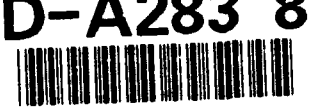
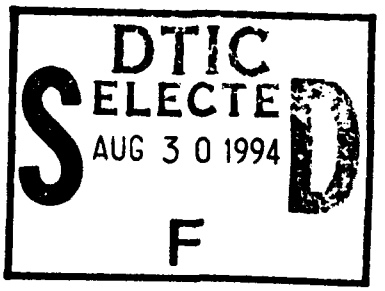
AUGUST 1994



UILU-ENG-94-2228
CRHC-94-13

(1)

Center for Reliable and High-Performance Computing



A Gate Level Simulator for Alpha-Particle-Induced Transient Faults

Hungse Cha

(10)

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CRHC-94-13	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1308 W. Main ST. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Joint Services Electronics Program 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014- 90-J-1270
8c. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A Gate Level Simulator for Alpha-Particle-Induced Transient Faults

12. PERSONAL AUTHOR(S) CHA, Hungse

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	1994 August 22	103

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	transient faults, alpha-particle-induced, VLSI circuits, speed

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Mixed analog and digital mode simulators have been available for accurate alpha-particle-induced transient fault simulation. Hoever, they are not fast enough to simulate a large number of transient faults on a relatively large circuit in a reasonable amount of time. This thesis describes a fast transient fault simulator which can evaluate the effects of alpha-particle hits or single event upsets (SEUs) in CMOS standard cell based synchronous sequential VLSI circuits. The speed comes from approximating the intial analog effects with gate level models, as well as using an improved transient fault simulation algorithm in a hierarchy of simulators. The simulator is shown to be between four to five orders of magnitude faster than a very accurate circuit simulator at the expense of some accuracy and some limitations on the types of circuits simulatable.

Using this simulator, benchmark circuits have been tested for their behavior under alpha-particle injections. The experiment show that the one bit flip model is not a good model for injecting fdaults in highly fault tolerant systems. The experiments also show that at the pin level, no simple model exists which can mimic the behavior of the circuit hit with the alpha particle.

The simulator's usefulness is also shown in the development of a transient pulse tolerant D flip-flop (DFF). The tool is used to demonstrate the tradeoff between transient pulse tolerance and latch performance.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

A GATE LEVEL SIMULATOR FOR ALPHA-PARTICLE-INDUCED TRANSIENT FAULTS

BY

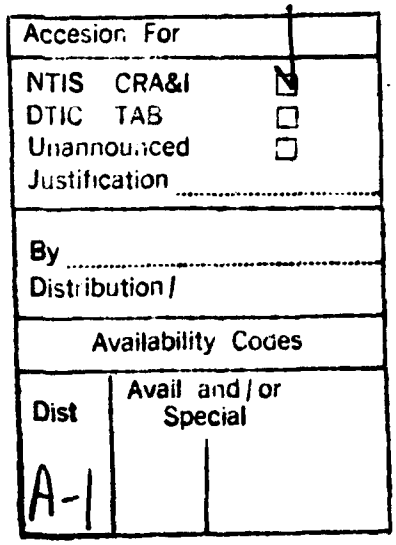
HUNGSE CHA

B.S., California Institute of Technology, 1988
M.S., University of Illinois at Urbana-Champaign, 1990

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1994

Urbana, Illinois



To my parents and my family

A GATE LEVEL SIMULATOR FOR ALPHA-PARTICLE-INDUCED TRANSIENT FAULTS

Hungse Cha, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1994
Janak H. Patel, Advisor

Mixed analog and digital mode simulators have been available for accurate alpha-particle-induced transient fault simulation. However, they are not fast enough to simulate a large number of transient faults on a relatively large circuit in a reasonable amount of time. This thesis describes a fast transient fault simulator which can evaluate the effects of alpha-particle hits or single event upsets (SEUs) in CMOS standard cell based synchronous sequential VLSI circuits. The speed comes from approximating the initial analog effects with gate level models, as well as using an improved transient fault simulation algorithm in a hierarchy of simulators. The simulator is shown to be between four to five orders of magnitude faster than a very accurate circuit simulator at the expense of some accuracy and some limitations on the types of circuits simulatable.

Using this simulator, benchmark circuits have been tested for their behavior under alpha-particle injections. The experiments show that the one bit flip model is not a good model for injecting faults in highly fault tolerant systems. The experiments also show that at the pin level, no simple model exists which can mimic the behavior of the circuit hit with alpha particles.

The simulator's usefulness is also shown in the development of a transient pulse tolerant D flip-flop (DFF). The tool is used to demonstrate the tradeoff between transient pulse tolerance and latch performance.

ACKNOWLEDGMENTS

I want to thank first and foremost my advisor Professor Janak Patel who has made this thesis possible with his insight, guidance and patience. I am grateful to Professors Prithviraj Banerjee, Ravi Iyer, Sung Mo Kang and Resve Saleh for serving on my doctoral committee. I want to express special gratitude to my parents for challenging me to rise above the comfort and security of mediocrity. I want to thank my wife, Soon-Ok Ha, for her support and patience throughout this project. I would like to thank Steven Parkes, Vivek Chickermane, Liz Rudnick, Jonathan Simonson, Abhijit Dharchoudhury, Han Seok Kim, Gwan Choi, Un-Ku Moon, Jong Won Shon, Inhwan Lee, Sungho Kim, Edward Chang, Jeff Rearick, Gary Greenstein, Michael Hsiao, Terry Lee, and Andy Hull, as well as other people in the Center for High Performance Computing and the Digital and Analog Circuits Group for their help and their friendship. Finally, I would like to thank JSEP for funding this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In today's complex and highly automated society, computers are increasingly being called upon to perform critical tasks where a failure could mean the loss of life. Inarguably, these computers should be very reliable and dependable.

The reliability of computers can be increased by the use of one or more fault tolerant schemes. Such schemes involve employing temporal or spatial redundancy so that the computer can continue to function correctly in the presence of one or more faults.

In the design phase of the fault-tolerant computers, different fault-tolerant schemes should be analyzed in order to assess their cost effectiveness as well as the overall level of reliability of the finished system. Typically, the computer system under design is injected with faults, and subsequent system behavior is analyzed to determine the above parameters. Fault injection may be performed either in actual hardware or in software through simulation. The former requires a prototype, making the technique expensive and time-consuming. The latter is preferred not only because it is less expensive but also because the analysis can be done early in the design phase to limit or eliminate costly redesign, thus, speeding the delivery of the finished product.

There are two types of faults that have to be considered for fault injection experiments on VLSI circuits: permanent and transient. Examples of permanent faults are stuck closed transistors or open metal lines. They can occur either during the fabrication process due to dust, or during normal operation due to aging processes such as hot carrier effects and metal migration. Transient faults, on the other hand, are faults deposited on the VLSI circuit which change the state of the circuit without harming the circuit itself. Thus, if the circuit were reset after a transient fault, it would begin to function correctly. Some sources of transient faults are alpha particles, electromagnetic interference, power supply variations, crosstalk, and lightning.

This thesis is concerned with the transient faults arising from the combinational part of a sequential circuit for two reasons: one, it is more difficult to simulate the injection of such a transient fault than that of a permanent fault since the former involves timing whereas the latter does not, and, two, it is an important problem since studies indicate that 85% or more of computer failures are transient in nature [1, 2]; this type of transient fault is a part of the cause.

Of the various sources of transient faults, the best understood and the most widely studied one is the alpha particle [3-11]. Alpha particles are mainly found in space, but small amounts of radioactive elements in the packaging material or solder can decay, resulting in alpha particles. The simulation of VLSI circuits under an alpha-particle hit is the main subject of this study. However, the simulation techniques that have been developed can be used for the simulation of transient faults caused by other sources as well, as long as an accurate model of the transient fault at the logic level can be developed.

The main reason for the difficulty in simulating the injection of a transient fault in the combinational logic portion is that the initial phenomenon is analog in nature, requiring an electrical level simulator for accuracy. Electrical level simulators such as SPICE [12] are extremely accurate but are also prohibitively slow for even moderately sized circuits. To make matters worse, there is a need to simulate a large number of fault injections to obtain statistically valid results due to the fact that a transient fault can occur randomly at any node in the circuit and at any time. The development of the simulator is made difficult by the dual requirements of accuracy and speed.

This thesis describes a fast transient fault simulator which can evaluate the effects of alpha-particle hits or single event upsets (SEUs) in CMOS standard cell based synchronous sequential VLSI circuits. An overview of the transient fault simulation environment is shown in Figure 1.1. The speed comes from approximating the initial analog effects with gate level models, as well as using an improved transient fault simulation algorithm in a hierarchy of simulators. The simulator TIFAS is shown to be between four to five orders of magnitude faster than SPICE at the expense of some accuracy and some limitations on the types of circuits simulatable.

As Figure 1.1 shows, the transient fault simulator (FAST) is actually composed of two separate simulators, TIFAS and TPROOFS [13]. TIFAS is a gate level event-driven simulator with assignable delay whereas TPROOFS is a modified version of PROOFS [14], a zero-delay parallel fault simulator. TIFAS uses the gate level models of the electrical level phenomena [15] to inject transients and to propagate their effects to the flip-flops in the circuits. Once a transient fault is latched, the remaining simulation is

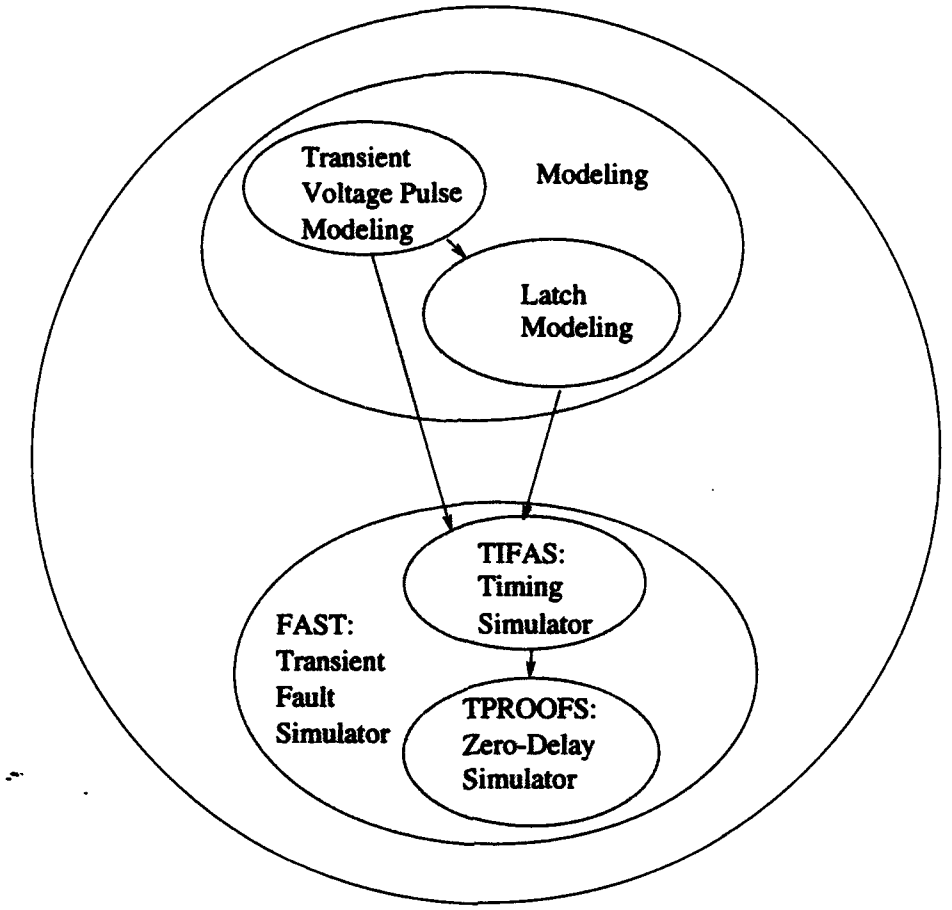


Figure 1.1 Overview of the transient fault simulator

performed using the faster TPROOFS since the timing details initially required are no longer needed.

This thesis describes the implementation details of the simulator as well as some experimental results on benchmark circuits obtained using the simulator. In Chapter 2, previous work in the area of fault injection is described. Chapter 3 describes the different models developed and used in the simulator. The implementation details of the simulator are discussed in Chapter 4. In Chapter 5, the accuracy of the simulator is discussed, comparing it to SPICE and ILLIADS [16]. In Chapter 6, fault injection experimental

results on benchmark circuits are presented. Chapter 7 describes an exercise in the use of FAST in designing DFFs to tolerate transient pulses. Finally, conclusions and future research are stated in Chapter 8.

CHAPTER 2

RELATED WORK IN FAULT INJECTION

There has been a significant amount of research done in the field of fault injection. Fault injection is necessary to study not only transient fault phenomena but also permanent fault phenomena, and the ability of fault-tolerant systems to deal with them. The fault injection experiments can be broadly categorized as software-implemented, hardware-implemented, or radiation-induced. In software-implemented fault injection, faults are injected in either software running on a system or a software simulation of some hardware. In hardware-implemented fault injection, faults are injected at the pins of the actual hardware, and in radiation-induced fault injection, heavy ions are used to inject charges. A more comprehensive survey of the related research in this area can be found in [17].

2.1 Radiation-Induced Fault Injection

Many studies have been done on the effects of high energy ions on circuits. The alpha particle as a source of transient upset in dynamic RAMs was first reported in [3]. The Z-80 and NSC-800 microprocessors are studied for their vulnerability to high

energy ions in [18]. The study of another microprocessor, MC6809E, under heavy-ion radiation is reported in [19]. Heavy-ion radiation is used to evaluate the effectiveness of error detection schemes in [20]. Data are gathered on the cosmic ray induced single event upsets on a computer in actual space environment in [21]. Although heavy ion experiments provide perhaps the most accurate data on a given circuit, they should be used as the final experiment prior to deployment. They are unsuitable at the early design phase for the obvious reason that the prototype is not available. Also, they have the drawback that the fault injection points are not controllable.

2.2 Hardware-Implemented Fault Injection

Other researchers have tested systems by injecting faults at the pin level. One of the first works in this area is described in [22] where faults are injected at the pin level on the FTMP (Fault-Tolerant Multiprocessor) to study its fault detection, isolation and recovery characteristics. To study the effectiveness of an on-line error detection mechanism, pin-level faults are injected on an MC68000 microprocessor in [23]. Pin-level fault injection experiments are conducted to validate the fault tolerance of a railway controller in [24]. Since pin-level fault injection in hardware also requires a prototype, this type of experiments is also an unsuitable method at the early design phase. Furthermore, there are only a limited number of fault injection sites since the faults are injected on the pins only.

2.3 Software-Implemented Fault Injection

There are several levels of software-implemented fault injection. At the highest level are flipping memory bits while an actual software is running on a hardware platform to mimic transient faults. This type of experiment is useful for studying the behavior of the system under transient faults. For example, in [25], a random page in memory is set to hexadecimal FF to study large system failures. In another work, faults are injected by flipping memory locations in IBM RT PCs [26]. Of course, the concern with the high-level fault injections is that the transient fault model used may not accurately reflect the actual fault phenomenon.

At a lower level, faults can be injected by flipping latches in HDL simulations of actual hardware. In [27], faults are injected at the flip-flops in HDL simulation of the TI SBR9000 microprocessor to study microprocessor program flow under single-event upsets. In [28], flip-flop faults are injected into an HDL simulation of a 32-bit RISC microprocessor. The same concern with the accuracy of the fault model exists here as well.

At an even lower level, fault injection experiments can be performed at the gate level. Transient gate-level faults are injected in a simulation model of the IBM RT PC in [29]. In that work, single-stuck-line transient faults lasting for one machine cycle were injected. The inaccuracy in this experiment is that the fault duration of one machine cycle is too long for alpha-particle injections, and it may be too long for other sources of transient faults as well.

To accurately simulate a fault injection, an electrical level simulator such as SPICE [12] with an appropriate model for alpha-particle injection [30, 31, 32] has to be used because the initial transient fault phenomenon is analog in nature [33, 34]. However, due to the long simulation times required by SPICE on even medium sized circuits, mixed-analog-digital-mode simulators have been developed to speed up the simulation time with little loss in accuracy [35, 36]. Researchers have used these mixed-mode simulators to inject transient faults into relatively large circuits [37, 38, 39]. However, even these simulators are slow when injecting a large number of faults into a large circuit.

Recently, a switch level timing simulator, ILLIADS [40], has been modified to inject transient faults due to alpha particles [16]. This work represents considerable improvement in speed over those for SPICE and mixed-mode simulators. As shown in Chapter 5, the simulator presented in this thesis is between three to four orders of magnitude faster than ILLIADS with some loss in accuracy for the simulation of standard cell based fully static CMOS synchronous sequential circuits. Undoubtedly, there will always be a place for ILLIADS and SPICE because of their accuracy, but for a quick look at the circuit behavior under alpha-particle injections, the simulator presented in this thesis would suffice.

CHAPTER 3

LOGIC LEVEL MODELING

Because an alpha-particle hit first manifests itself at the electrical level, a logic level model of its effects has to be developed to enable gate-level transient fault simulation; the logic-level model is the subject of this chapter. Since the inverter is the easiest gate to understand and since other gates can be modeled as inverters, only inverters will be considered for the most part in this chapter. First, the immediate effects of alpha-particle hit and the detailed logic-level model are presented. In this presentation, the rise/fall delays of gates are also discussed. Then, a flip-flop model in the presence of transient voltage pulses is presented. Finally, the actual models used in the transient fault simulator are described.

3.1 The Immediate Effects of Alpha Particles

Researchers have extensively studied the phenomenon of alpha particles striking MOS devices, starting with the seminal works by May and Woods [3] and by Messenger [31]. An alpha-particle hit on a semiconductor creates electron-hole pairs in its track as it gives up energy. The electron-hole pairs thus created will first diffuse and then drift if

there is an electric field. If the alpha particle goes through any portion of a depletion region, the electron-hole pairs created will drift due to the electric field in the region. The drifting phenomenon causes charges to be collected across the depletion region. This charge collection can be modeled by a double-exponential current pulse [31]

$$I(t) = I_o(e^{-t/\tau_\alpha} - e^{-t/\tau_\beta}), \tag{3.1}$$

where τ_α is the collection time constant of the junction and τ_β is the time constant for initially establishing the ion track. The time constants for the exponentials are functions of several fabrication process dependent factors, and in this thesis, the time constants given in [41] are used: $\tau_\alpha = 1.64 \times 10^{-10}$ sec and $\tau_\beta = 5.0 \times 10^{-11}$ sec. Figure 3.1 shows the current pulses described by Eq. (3.1) as a function of time and the total charge injected.

There are four possible cases of charge injection scenarios for CMOS circuits as shown in Figure 3.2, where the resistors represent conducting transistors and the direction of the arrows inside the current sources corresponds to the direction of the current flow. For cases I and II, the voltage at the p^+ node will temporarily rise up, and for cases III and IV, the voltage at the n^+ node will momentarily fall down. Cases II and IV will not affect the logic state of the circuit because the node is already at the logic value toward which the injected charge will drive the node. However, cases I and III may affect the logic value of the node, which, in turn, may cause incorrect operation of the circuit.

Consider the inverter shown in Figure 3.3. If the input of the inverter is held constant during charge injection and the subsequent voltage pulse production at the output, we

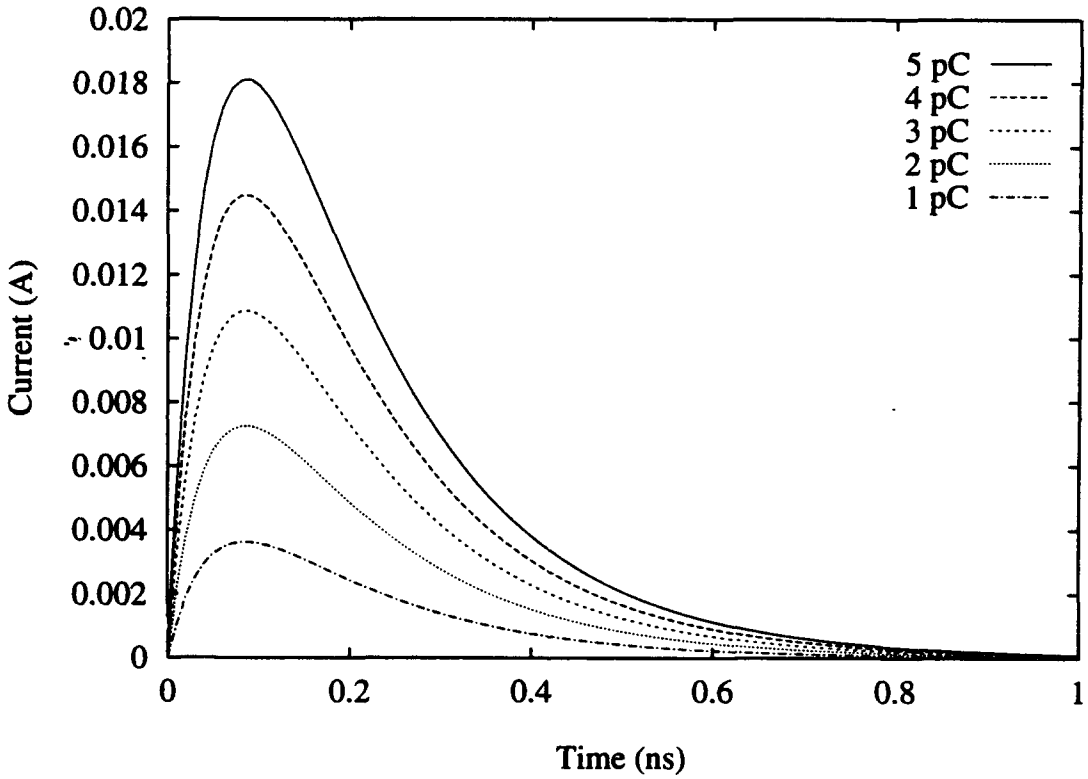


Figure 3.1 Current pulses resulting from alpha-particle hits

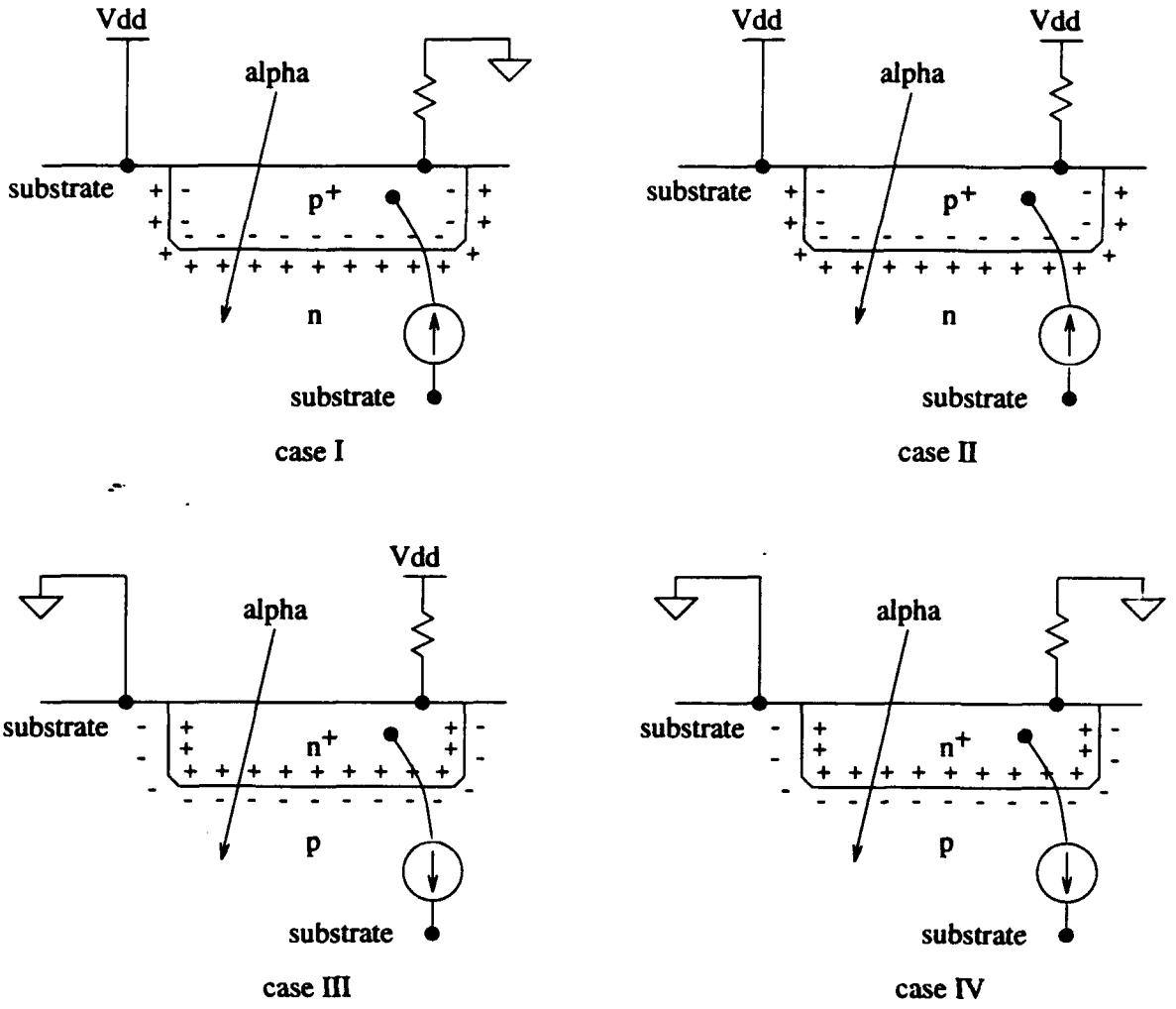


Figure 3.2 Four possible scenarios for charge injection due to alpha particles

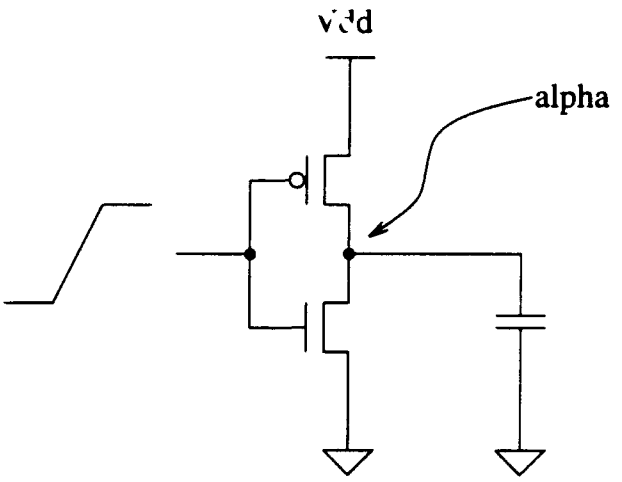


Figure 3.3 Alpha-particle hit on an inverter with transitioning input

will have one of the four cases in Figure 3.2. However, if the input is transitioning, we will either have a combination of cases I and II, or a combination of cases III and IV. Suppose that the input of the inverter is transitioning from ground to V_{DD}. If the alpha particle hits the drain of the NMOS transistor, we have a combination of cases III and IV and the output node will be driven to logic 0. Since the output is already being driven to logic 0 by the input, the alpha particle will actually aid in the output transition, and this hit will not manifest itself as a fault. We have a more interesting scenario when the alpha particle hits the drain of the PMOS transistor with the same input transition. This is a combination of cases I and II depending on when the alpha particle strikes relative to the input and output transitions. As mentioned above, case II does not cause an erroneous logic behavior. It has been found that transient faults injected before a node has settled to its final logic value for the clock cycle rarely, if at all, manifest themselves as latched errors [13]. Therefore, no effort was made to model the effects of an alpha particle that is injected during a node transition. Instead, this scenario is considered as case I, and

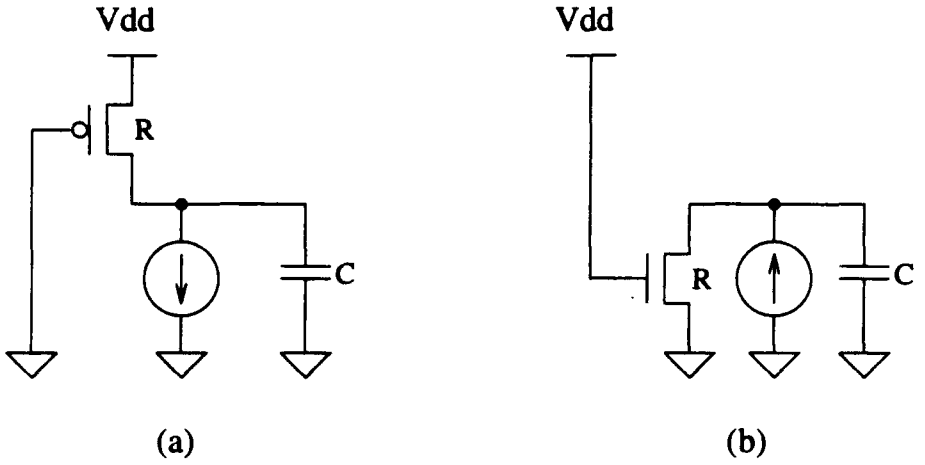


Figure 3.4 Equivalent circuit of an inverter with the input tied to ground or V_{DD}

the complementary scenario in which the input is transitioning from V_{DD} to ground is considered as case III.

3.2 The Pulse Width at the Injection Node

The circuit under an alpha-particle hit can be viewed as an inverter as shown in Figure 3.3 with the input tied to either ground or V_{DD}, which leads to the equivalent circuits of Figures 3.4 (a) and (b), respectively, with the current source specified by Eq. (3.1). Here, the discussion will be focused on the case shown in Figure 3.4 (a) since the case shown in (b) can be analyzed in a similar manner.

When an alpha particle hits the drain of the NMOS transistor, the voltage at the injection node will temporarily dip down and then rise back up as the PMOS transistor supplies current to recharge the node. The voltage function depends on three quantities: strength of the PMOS transistor, the injected charge, and the total capacitance at the

injection node. If we view the PMOS transistor as a linear resistor, we have an RC circuit with the injected current modeled by Eq. (3.1). This circuit can be solved analytically for the voltage at the output node as a function of time, and the solution is found to be

$$V_o(t) = V_{DD} + RI_o \left[\frac{e^{-t/\tau_\alpha} - e^{-t/RC}}{1 - RC/\tau_\alpha} - \frac{e^{t/\tau_\beta} - e^{-t/RC}}{1 - RC/\tau_\beta} \right]. \tag{3.2}$$

A transistor is a highly nonlinear device and can not be modeled as a linear resistor, but its resistance is proportional to its L/W ratio. Upon examining Eq. (3.2), we find that $V_o(t)$ is a function of two quantities, RI_o and RC. This observation potentially simplifies our model of the voltage pulse width because now there is a possibility that it could be described as a function of two quantities instead of three. In fact, SPICE simulation verifies that it can indeed be modeled with two quantities.

For the development of the models, the SPICE level-two parameters from Orbit, a fabrication company accessible through MOSIS, have been used for the MOS transistors. The inverter design used is from the standard cell library from Mississippi State University, which is distributed as part of the OCTTOOLS [42] set from the University of California, Berkeley. SPICE3 [43] simulations have been run on the *invf101* inverter with the injected charge ranging from 1 pC to 9 pC and the output driving between 1 and 31 *invf101* inverters. The voltage waveforms produced at the injection node are then processed to give logic level pulses with the logic threshold set at 2.5 V. Figure 3.5 shows a sample waveform. The pulse width as a function of the injection charge from 4 pC to 9 pC and the number of fanout inverters from 1 to 31 is shown in Figure 3.6. In the figure, we can immediately see two distinct regions of pulse width behavior. For

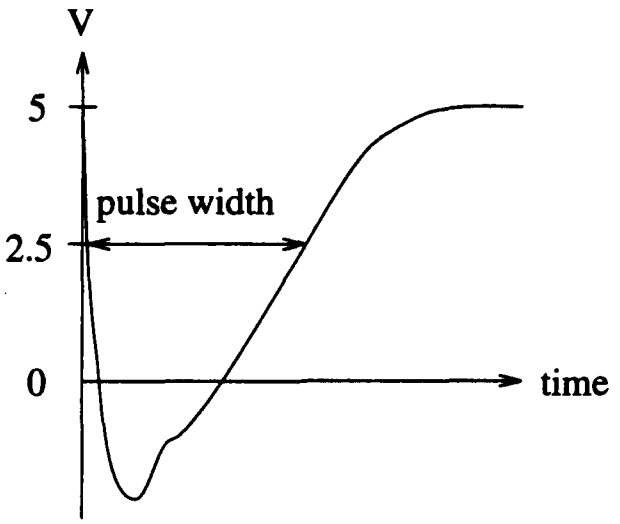


Figure 3.5 Sample voltage waveform at the charge injected node

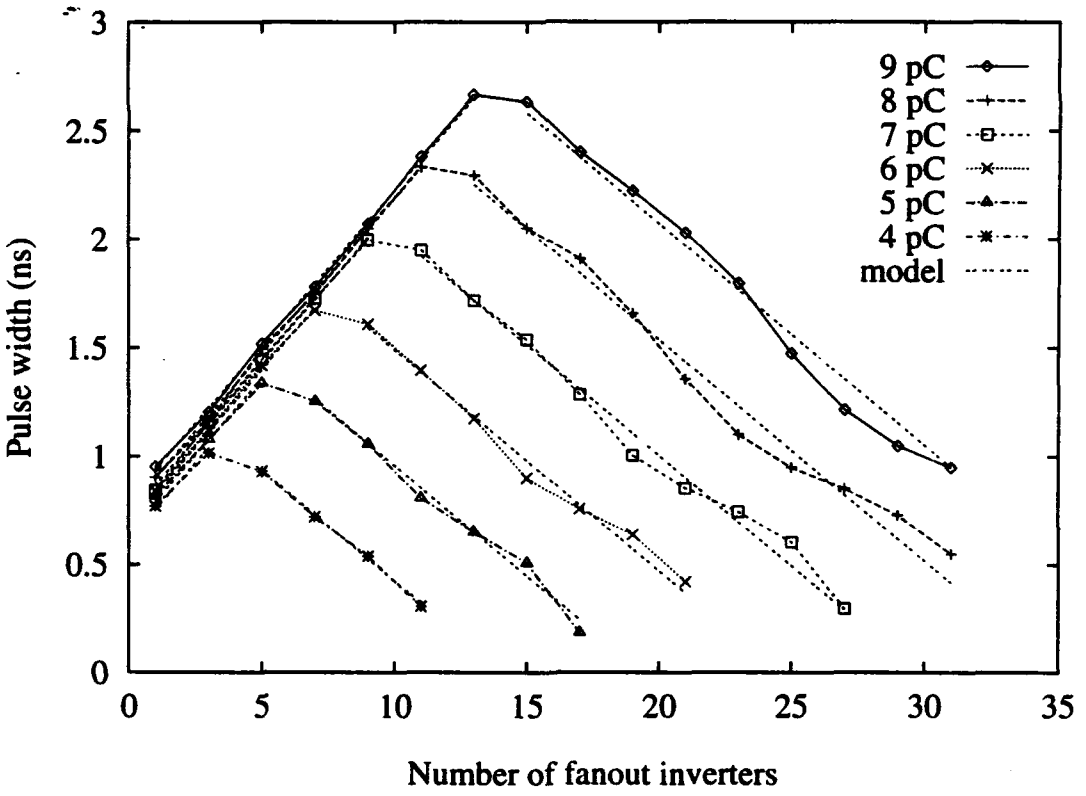


Figure 3.6 Pulse width at the injected node

a smaller number of fanout inverters, the pulse width becomes larger as the number of inverters increases. This is due to the slower RC time constant for recharging the node. For large capacitances, the injected charge is not sufficient to drive the voltage at the node all the way down to ground. Therefore, as the output capacitance increases, the voltage dips less and less, resulting in smaller pulse width.

The two distinct regions can be modeled with two linear equations of the form

$$PW_1 = A\frac{L}{W}I_o + B\frac{L}{W}C_{out} + Const, \tag{3.3}$$

where A, B, and $Const$ are obtained from linear regression analysis of SPICE3 data, and PW_1 stands for the pulse width at the injection node. The PW_1 computed using the above equation is superimposed on the SPICE3 data in Figure 3.6. As can be seen, the model and SPICE3 data agree quite well.

3.3 Propagation Delay of Inverters

Researchers have found that at least three gate levels as shown in Figure 3.7 are needed in SPICE-like simulations until the electrical effects become stable enough to be treated as logic signals [37, 38]. However, most of the time the electrical effects become stable enough to be treated as logic signals after only two gate levels and the additional effort involved in modeling the third gate level is not justified.

In the previous section, a simple model for the pulse width at the injection node has been developed. This model, however, does not give us any information about the shape of the pulse, which is needed since the gate delays are functions of the input slew rates.

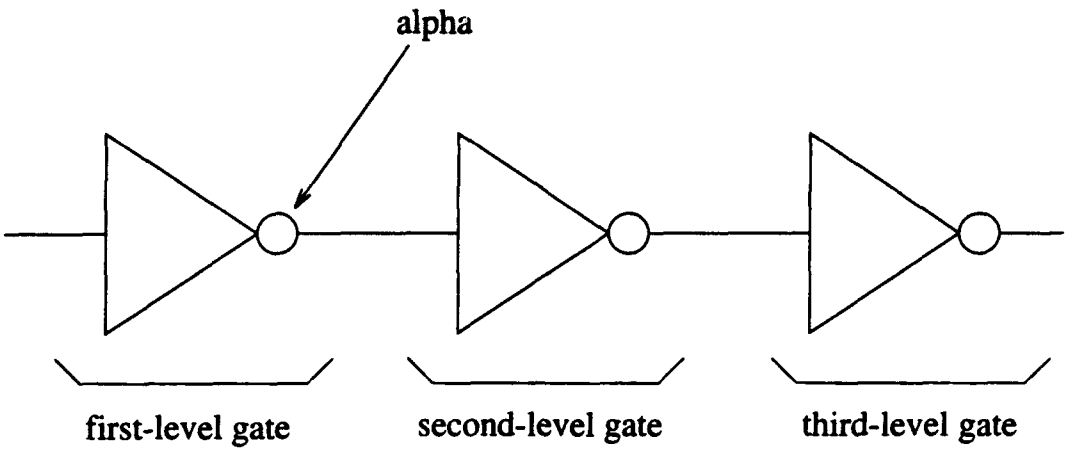


Figure 3.7 Definition of gate levels

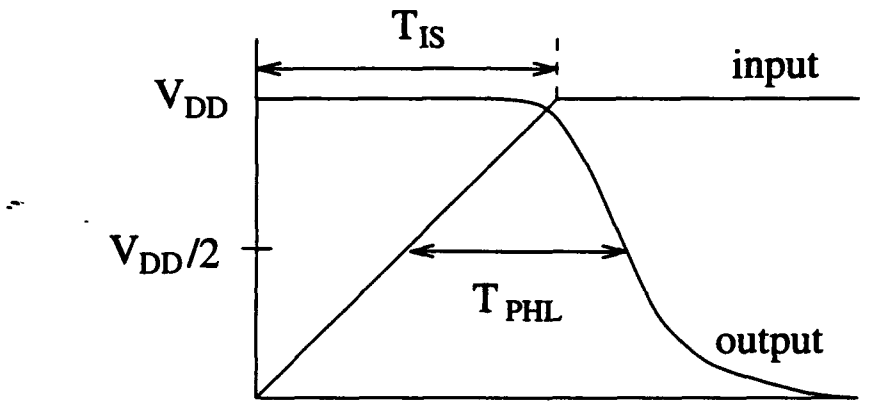


Figure 3.8 Definition of propagation delay of inverters

Once the shape of the pulse is known, the pulse width at the output of a second-level gate can be modeled. The propagation delay of gates is necessary for this purpose.

The propagation delay is defined as the time it takes for the output to reach $V_{DD}/2$ minus the time it takes for the input to reach $V_{DD}/2$. The falling delay τ_{PHL} is shown in Figure 3.8, and the rising delay τ_{PLH} is similarly defined. SPICE3 simulations of *invf101* with various values for T_{IS} are shown in Figure 3.9. As can be seen in the figure, the propagation delay is a nonlinear function of the output capacitance and T_{IS}. Therefore,

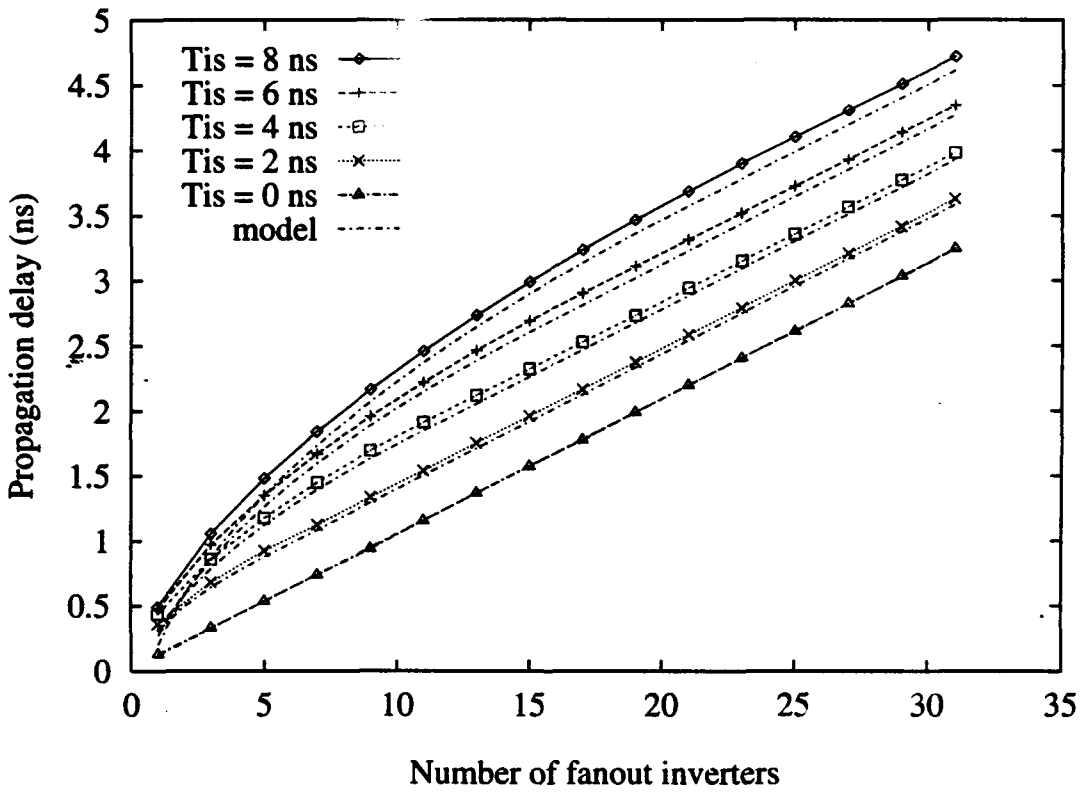


Figure 3.9 Propagation delay of the inverter as a function of fanout loading for various values of T_{IS}

a simple curve-fitting technique is inappropriate, and a deeper insight into the behavior of the propagation delay as a function of these parameters is required.

With the following equation for the NMOS transistor,

$$I_D = \begin{cases} k'_N(W/L)_N[(V_{in} - V_{TN})V_{out} - V_{out}^2/2] & \text{if } V_{in} - V_{TN} > V_{out} \\ \\ k'_N(W/L)_N(V_{in} - V_{TN})^2 & \text{if } V_{in} - V_{TN} < V_{out} \end{cases} \tag{3.4}$$

and a similar one for the PMOS transistor, the step response of an inverter can be analyzed. We concentrate on τ_{PHL} as the analysis can easily be extended to τ_{PLH} by switching the roles of the PMOS and the NMOS transistors.

A closed-form solution for the propagation delay is given in [44]

$$\tau_{PHL} = D \frac{L}{W} C_{out} \tag{3.5}$$

where D is a process-dependent parameter. Although this solution was obtained using the square law current equations for the transistors as given above, it can be applied in general with the parameter D extracted from SPICE3 simulations followed by linear regression analysis.

The above equation is fine for step input change, but as can be seen in Figure 3.9, the propagation delay is also a function of the input slew rate. There has been a significant amount of research done to model τ_{PHL} as a function of ramping inputs [45, 46, 47, 48]. Of these, only Shoji's work [47] is simple enough to be used in a logic level simulator, and his analysis is discussed below.

To obtain a closed-form solution, he has used very simple equations for the current of the transistors. For the PMOS transistor,

$$I_P = b_P\,(V_{DD} - V_G) \;\text{ if } V_D < V_{DD}, \tag{3.6}$$

and current less than I_P can flow when $V_D = V_{DD}$. For the NMOS transistor,

$$I_N = b_N\,V_G \;\text{ if } V_D > 0, \tag{3.7}$$

and current less than I_N can flow if $V_D = 0$. The parameters b_P and b_N may be thought of as transconductance parameters of the PMOS and the NMOS transistors, respectively. The gate voltage is described as

$$V_G(t) = \begin{cases} 0 & \text{if } t < 0 \\ \alpha t & \text{if } 0 < t < T_{IS} \\ V_{DD} & \text{if } t > T_{IS}, \end{cases} \tag{3.8}$$

where $\alpha = V_{DD}/T_{IS}$ and T_{IS} is the time it takes for the input voltage to slew from 0 V to V_{DD}. Solving for the propagation delay, Shoji obtains

$$\frac{\tau_{PHL}}{\tau_{PHL\infty}} = \begin{cases} 2\sqrt{\frac{\alpha_0}{\alpha}} + \frac{1-\beta}{\beta}\frac{\alpha_0}{\alpha} & \text{if } \alpha < \alpha_0 \\ \\ 1 + \frac{1}{\beta}\frac{\alpha_0}{\alpha} & \text{if } \alpha > \alpha_0, \end{cases} \tag{3.9}$$

where $\beta = b_N/b_P$, $\tau_{PHL\infty}$ is the τ_{PHL} for the step function or the infinite slew rate input, and

$$\alpha_0 = \frac{\beta}{2\,(1+\beta)}\,\frac{V_{DD}}{\tau_{PHL\infty}}. \tag{3.10}$$

Since these equations were derived from very simple current equations for the transistors, we can not expect these equations to accurately describe the propagation delay. However,

we can expect them to provide us with an insight into the first-order behavior of the propagation delay. We can rewrite the equation for the region $\alpha > \alpha_0$ as

$$\tau_{PHL} = \tau_{PHL\infty} + \frac{1}{2(1+\beta)} T_{IS}. \tag{3.11}$$

This equation has been modified as

$$\tau_{PHL} = \tau_{PHL\infty} + (E + F/\beta) T_{IS}, \tag{3.12}$$

where E and F are parameters to be extracted from SPICE simulations followed by curve fitting. An effort has been made to change Eq. (3.11) as little as possible yet introduce enough parameters to accurately model the realistic propagation delay. Armed with Eq. (3.12), we may write

$$\frac{\tau_{PHL}}{\tau_{PHL\infty}} = \begin{cases} 2\sqrt{\frac{\alpha'_0}{\alpha}} + \frac{1-\beta'}{\beta'}\frac{\alpha'_0}{\alpha} & \text{if } \alpha < \alpha'_0 \\ \\ 1 + \frac{1}{\beta'}\frac{\alpha'_0}{\alpha} & \text{if } \alpha > \alpha'_0 \end{cases} \tag{3.13}$$

where

$$\beta' = \frac{1}{G}\frac{k'_N(W/L)_N}{k'_P(W/L)_P} \tag{3.14}$$

and

$$\alpha'_0 = \frac{\frac{k'_N(W/L)_N}{k'_P(W/L)_P}E + F}{G}\frac{V_{DD}}{\tau_{PHL\infty}}. \tag{3.15}$$

Note that Eq. (3.13) is exactly the same form as Eq. (3.9). The parameter G has been introduced to provide better matching of the model and the actual data in the region $\alpha < \alpha'_0$. This parameter is also to be extracted from SPICE simulations followed by curve fitting. The introduction of this parameter has no effect on the equation in region

$\alpha > \alpha_0'$ since the definitions of β' and α_0' cancel out this factor. The computed values of τ_{PHL} from the above equations are superimposed on the data from SPICE3 simulations in Figure 3.9. The matching between the model and the SPICE data is very good, as can be seen in the figure.

3.4 At the Output of a Fanout Inverter

The transient pulse at the output of the second-level gate has two quantities of interest: the width of the pulse, and the delay through the second-level gate. To find these quantities, we first have to characterize the shape of the voltage pulse at the injection node, which can be done using the models developed in the previous sections. Consider case III in Figure 3.2. Upon current injection, the voltage at the injection node will drop rapidly to 0 V or below. Since the time constants for charge injection are very small, this rapid drop will occur relatively independent of the capacitance and the strength of the PMOS transistor at the injection node. Therefore, this part of the voltage pulse is modeled as a step function going from V_{DD} to ground. The voltage stays at or below 0 V for a certain amount of time before it begins to rise as the PMOS transistor recharges the node. This portion of the pulse is modeled as being 0 V. When the voltage does begin to rise above 0 V, the charge injection can be assumed to have been completed since the time constants for the current pulse are on the order of 0.1 ns or less. If the charge injection has indeed been completed by the time the voltage starts to rise above 0 V, we have exactly the same conditions as the step input change. Therefore, from this point in

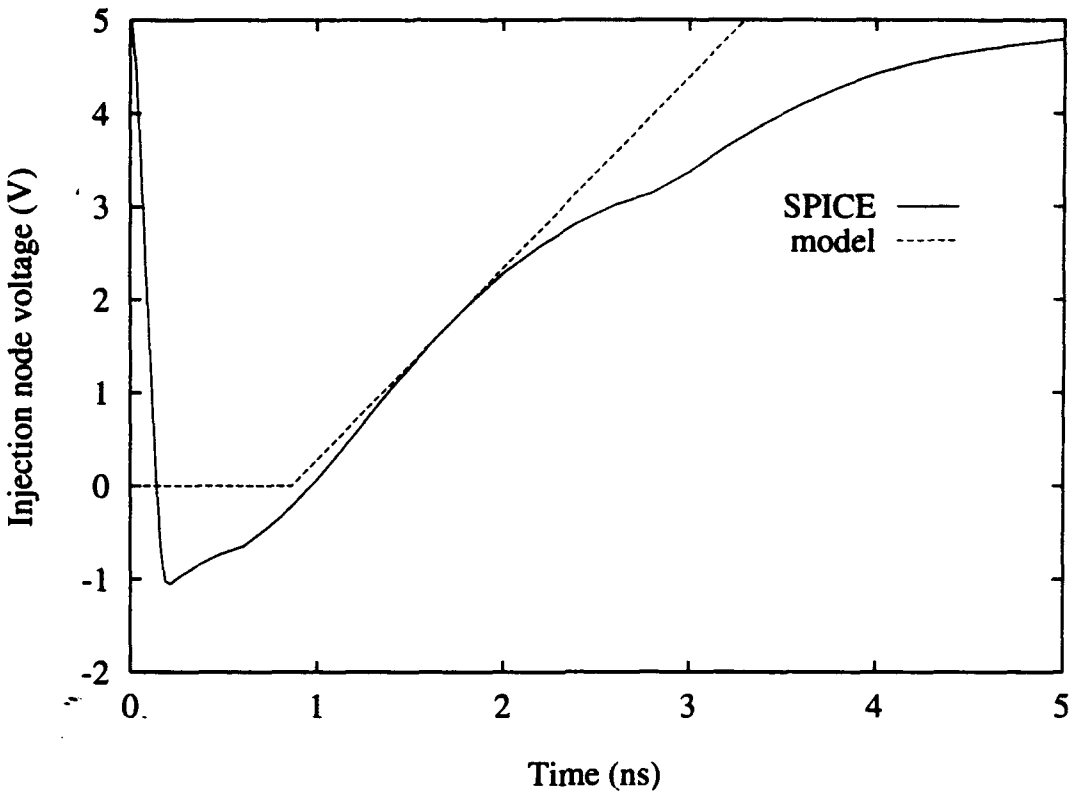


Figure 3.10 Model of the pulse waveform at the injection node

time, the injection node displays a step response as found in the previous section. The rising part of the pulse is modeled as a ramp with $T_{IS} = 2\,\tau_{PLH}$, and the pulse begins to rise at time $t = PW_1 - \tau_{PLH}$. Figure 3.10 shows the model of the pulse waveform adopted and the actual voltage waveform at the injection node.

Now we are ready to find the pulse width at the output of the second-level inverter. According to our model of the voltage pulse waveform at the injection node, the second-level inverter will see two transitions, one a step and the other a ramp. We can immedi-

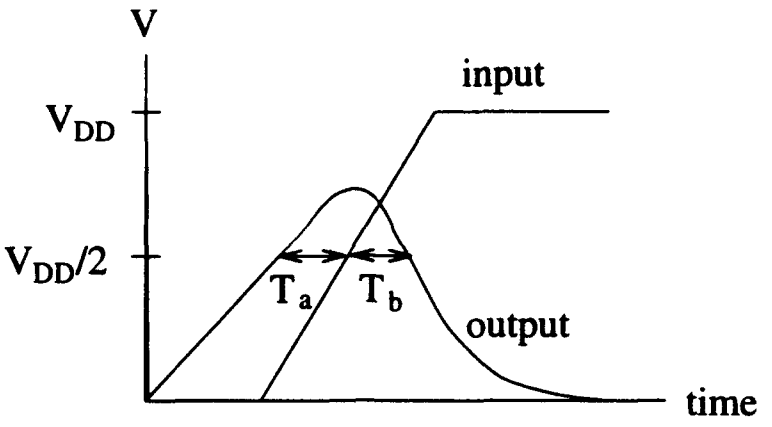


Figure 3.11 Slow transitioning of the inverter with respect to the input pulse

ately write the following equation to describe the pulse width of the second inverter:

$$PW_2 = \tau_{PHL2} - \tau_{PLH2} + PW_1. \qquad (3.16)$$

The subscript 1 refers to values at the injection node while the subscript 2 refers to values at the output of the second level inverter. The values τ_{PHL2} and τ_{PLH2} are functions of the input slew rate, capacitive loading, etc., and are computed accordingly. Equation (3.16) holds provided that (1) the output of the second level inverter has had enough time to rise to V_{DD} before it is pulled down to ground and (2) the injection node voltage drops down to ground before rising back up. If not, the equation for PW_2 becomes more complicated; we will discuss each of the cases in turn.

Figure 3.11 illustrates the first case. While the output of the second inverter is still transitioning due to the step input change, the input has risen sufficiently high to start driving the output to ground. Define T_a and T_b as seen in the figure, and assume that the output rises in a linear fashion until the input has risen to $V_{DD}/2$ at which time the

output starts falling, also in a linear fashion. Then we have

$$T_a = PW_1 - \tau_{PLH2} \tag{3.17}$$

and

$$T_b = \frac{2\tau_{PHL2}}{V_{DD}} \left(\frac{V_{DD}}{2\tau_{PLH2}} PW_1 - \frac{V_{DD}}{2} \right). \tag{3.18}$$

Combining Eqs. (3.16), (3.17), and (3.18) together, we have

$$PW_2 = \begin{cases} 0 & \text{if } \tau_{PLH2} > PW_1 \\ PW_1 - \tau_{PLH2} + \frac{2\tau_{PHL2}}{V_{DD}} \left(\frac{V_{DD}}{2\tau_{PLH2}} PW_1 - \frac{V_{DD}}{2} \right) & \text{if } \tau_{PLH2} < PW_1 < 2\tau_{PLH2} \\ \tau_{PHL2} - \tau_{PLH2} + PW_1 & \text{if } 2\tau_{PLH2} < PW_1 \end{cases} \tag{3.19}$$

The second case that has to be considered is when the input voltage does not fall all the way down to 0 V. In this case, the PMOS transistor is not fully turned on, and the NMOS transistor may be conducting if the input voltage is high enough.

Ignoring the NMOS transistor to simplify our analysis, we can perform the analysis done in [44] with a step change at the input whose final value is V_{IN}.

$$T_{PHL} = \tau_N \frac{1}{V_{in} - V_{TN}} \left[\frac{2(V_{DD} - V_{in} + V_{TN})}{V_{in} - V_{TN}} + \ln \left(\frac{4(V_{in} - V_{TN})}{V_{DD}} - 1 \right) \right] \tag{3.20}$$

and

$$T_{PLH} = \tau_P \frac{1}{V_{DD} - V_{in} - |V_{TP}|} \left[\frac{2(V_{in} + |V_{TP}|)}{V_{DD} - V_{in} - |V_{TP}|} + \ln \left(\frac{4(V_{DD} - V_{in} - |V_{TP}|)}{V_{DD}} - 1 \right) \right], \tag{3.21}$$

where $\tau_N = \frac{C_{out}}{k'_N(W/L)_N}$ and $\tau_P = \frac{C_{out}}{k'_P(W/L)_P}$. The propagation delay we use must be modified according to how much the input voltage has dropped with respect to some reference value. We will set the reference value to be the minimum input voltage for the maximum PW_1 obtained by varying the capacitance while holding the other parameters constant. Then,

$$V_{in_{min}} = \frac{V_{DD}}{2} - \frac{V_{DD}}{T_{IS}} PW_1 \tag{3.22}$$

and

$$V_{in_{min}}|_{PW_1=max} = \frac{V_{DD}}{2} - \frac{V_{DD}}{T_{IS}|_{PW_1=max}} PW_1|_{PW_1=max}. \tag{3.23}$$

The maximum value of PW_1 as well as $T_{IS}|_{PW_1=max}$ can be found directly from Eq. (3.3). Although the input voltage does not stay at the minimum value to which it has dropped, we will approximate the effect as such. Then, we have

$$\tau'_{PHL\infty} = \tau_{PHL\infty} \frac{factor_N(V_{in_{min}})}{factor_N(V_{in_{min}}|_{PW_1=max})} \tag{3.24}$$

where

$$factor_N(V_x) = \frac{1}{V_x - V_{TN}} \left[\frac{2(V_{DD} - V_x + V_{TN})}{V_x - V_{TN}} + \ln\left(\frac{4(V_x - V_{TN})}{V_{DD}} - 1 \right) \right]. \tag{3.25}$$

The new propagation delay $\tau'_{PHL\infty}$ is used in Eqs. (3.13) and (3.15) in place of $\tau_{PHL\infty}$ for the computation of τ_{PHL}.

Figure 3.12 shows PW_2 from SPICE3 simulations as a function of the injected charge and the capacitance at the injection node. The capacitance at the output of the second-level gate is equal to the gate capacitance of one inverter in this simulation run. The figure also shows the computed values of PW_2 from the model which incorporates all of

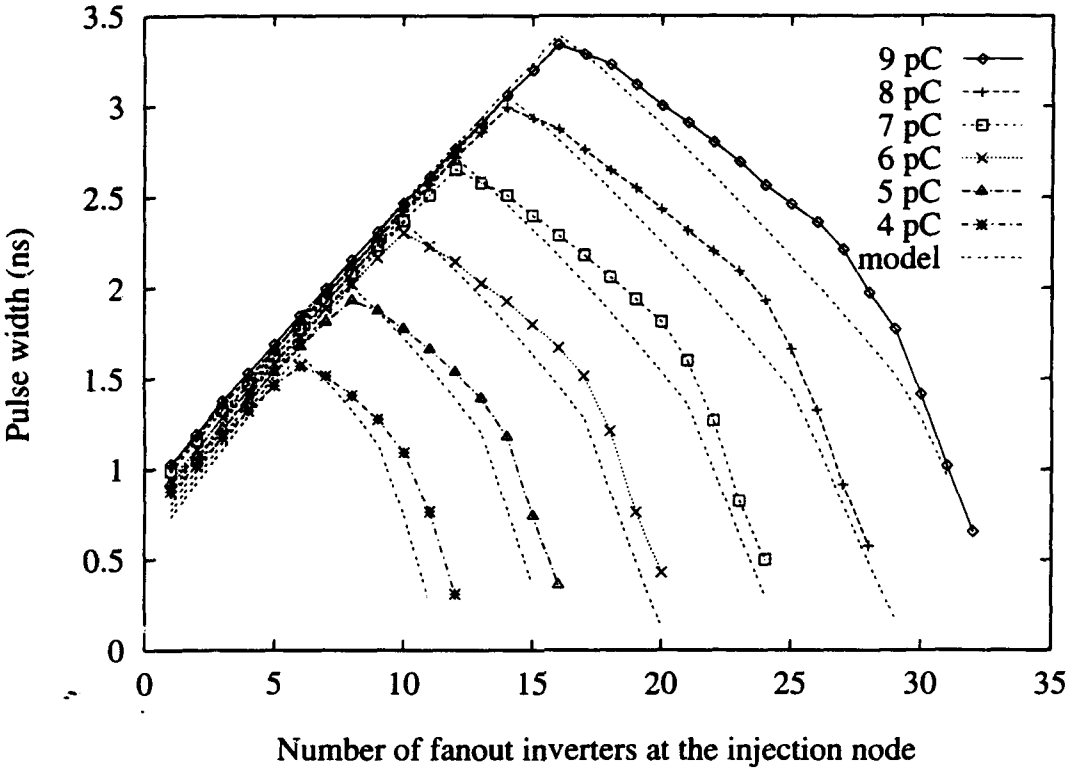


Figure 3.12 Pulse width at the output of the second inverter

the cases discussed above. Although there are some discrepancies, we see that the model tracks the SPICE data well.

The last bit of information needed at the output of the second-level inverter is the delay of the pulse since we are interested in the pulse waveform. This is easily obtained since it is $\tau'_{PLH\infty}$. Figure 3.13 shows the pulse delay data from SPICE simulations as well as the values computed from the model for the same set of simulations done for Figure 3.12. There are some discrepancies between the SPICE data and our model. However, considering all the simplifying assumptions that have been made in the development of

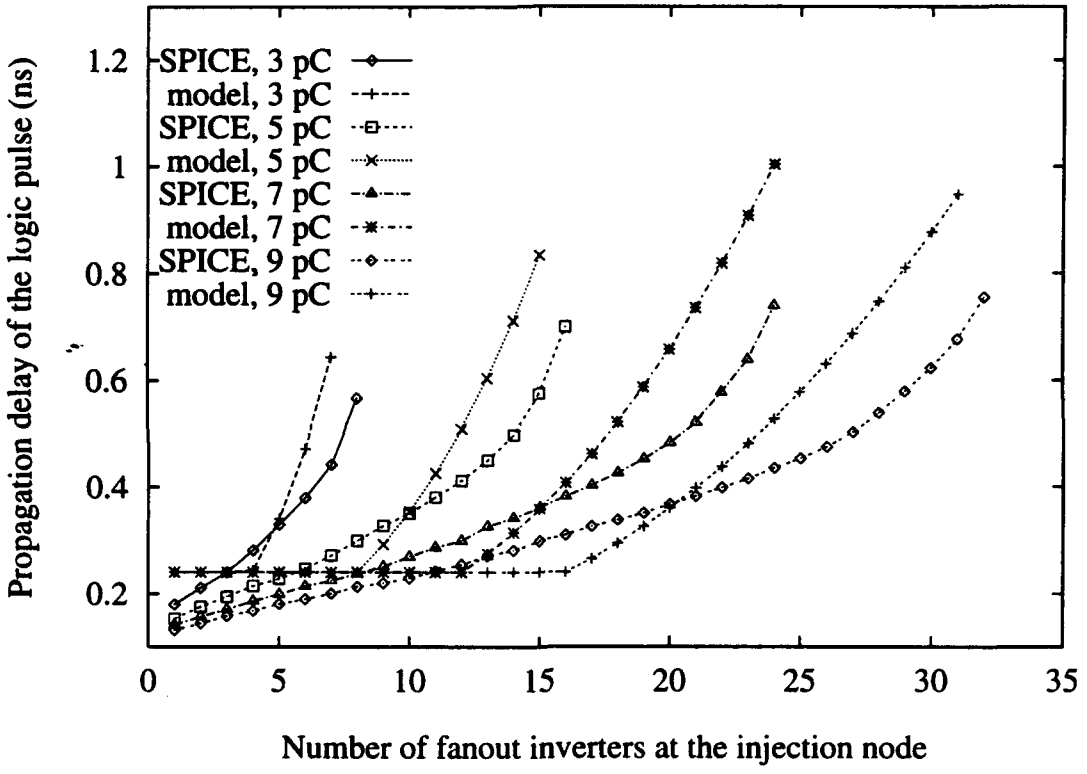


Figure 3.13 The delay of the transient pulse from the injected node to the output of the second level gate

our model and the simplicity of the model itself, the SPICE data and the model agree well.

3.5 Latch Modeling

All flip-flops have setup-time and hold-time specifications which are defined with respect to the clock edge. Sequential circuits are designed so that these times are satisfied. However, in the presence of transient pulses, there is no guarantee that the setup and hold times will be satisfied; furthermore, the interesting cases from a transient fault simulation standpoint are the ones in which they are violated. Therefore, a more detailed model of the behavior of the flip-flops with respect to pulses arriving in the neighborhood of the clock edge is needed.

To obtain such a model of the flip-flop behavior, SPICE simulations were performed on the *dfnf311* DFF in the standard cell library from Mississippi State University. This is a master-slave type D flip-flop which latches the input on the negative transition of the clock. The input and clock signals used are simple piecewise linear waveforms with rise and fall times of 0.5 ns as shown in Figure 3.14. The figure shows the two types of transient pulses which can occur as well as a clock transition. The DFF was simulated with various durations and arrival times of the input pulse in increments of 0.1 ns. The duration of the input pulse is defined as the interval from mid-point to mid-point of the two transitions in the pulse, as shown in Figure 3.14. The arrival time of a transient pulse is defined as the arrival time of the second transition of the pulse with respect to

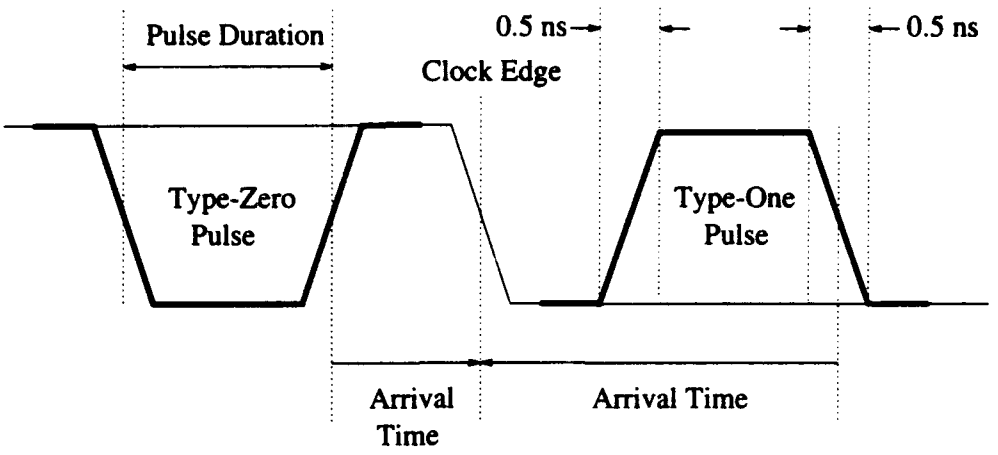


Figure 3.14 Transient pulses applied at the input of DFF

the clock edge. For example, in Figure 3.14, the type-zero pulse on the left has a negative arrival time and the type-one pulse on the right has a positive arrival time.

The simulations show that for a type-one pulse, pulses with durations less than 0.7 ns are not latched. Pulses with durations between 0.7 ns and 1.5 ns may be latched depending on their arrival times with respect to the clock edge. *Latching window* is defined as the neighborhood of the clock edge in which a pulse of given duration is latched, and generally, it is larger for wider pulses. The latching windows for various pulse durations for the type-one pulse are shown in Table 3.1. The **Earliest Time** and the **Latest Time** define the limits of the latching window as illustrated in Figure 3.15.

For this particular DFF, type-zero pulses of durations less than or equal to 1.5 ns are not latched because the *dfnf311* cell was designed with unbalanced PMOS and NMOS drive at the input, resulting in longer pulse durations to flip the latch from 1 to 0. In effect, this DFF is immune to flip-to-zero errors for the pulses widths less than or equal to 1.5 ns. For other DFF designs, both flip-to-one and flip-to-zero errors may occur and

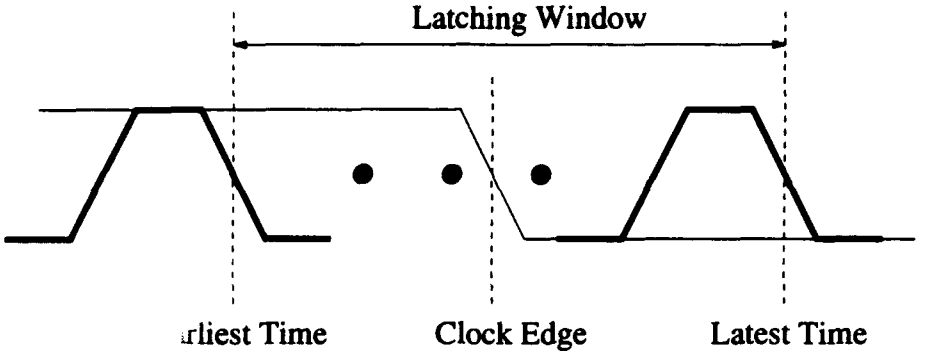


Figure 3.15 Neighborhood of clock edge in which the transient pulse is latched

possibly with different latching windows. For some of the experiments in the following chapters, the flip-to-zero latching window was made identical to the flip-to-one latching window in order to model a more general DFF behavior. For others, a DFF that has been modified to equalize the latching windows is used.

Table 3.1 DFF latching windows for various transient pulse durations

Pulse Duration *(ns)*	**Earliest Time** *(ns)*	**Latest Time** *(ns)*
0.7	0.2	1.1
0.8	-0.5	1.2
0.9	-0.8	1.3
1.0	-1.0	1.4
1.1	-1.1	1.5
1.2	-1.2	1.6
1.3	-1.3	1.7
1.4	-1.4	1.8
1.5	-1.5	1.9

The last type of modeling required is the behavior of the flip-flop when an alpha particle hits its output node. *Dfnf311*, as mentioned above, is a master-slave type flip-flop which latches the input on the falling transition of the clock. When the clock is

low, the slave latch receives its data from the master latch, so the output of the flip-flop acts like any other gate. However, when the clock is high, the feedback path in the slave latch is enabled, and charge injected on the output node may flip the state of this latch. The problem with a flipped latch is that the transient pulse width becomes much longer, and this phenomenon must be modeled correctly. For a given injected charge, there is a threshold output loading below which the latch will be flipped and over which the latch maintains the correct state. The threshold loading was found for total charges ranging from 1 pC to 9 pC in increments of 1 pC. If the output loading is below this threshold, the pulse is modeled to last until the next clock edge. If the output loading is over this threshold, the pulse is assumed to die out. Table 3.2 shows the threshold loadings for the slave latch flipping from 0 to 1.

Table 3.2 Dfnf311 threshold loading values for the slave latch flipping to 1

Charge (pC)	1	2	3	4	5	6	7	8	9
Threshold Loading (# inverters)	0	0	0	2	5	8	12	16	20

There is some concern that the latch might go into a metastable state [49, 50, 51, 52, 53] if the pulse arrives at exactly the right moment with respect to the clock edge. In this work, the latch metastability phenomenon was not observed in the SPICE simulations performed to model the latch; therefore, it has been ignored.

Finally, it should be noted that the modeling methodology presented in this chapter has some limitations. First, the pulse width modeling methodology is well suited for fully static CMOS standard cell design where there is only a limited number of different gates that have to be modeled. Second, the latch model is well suited for synchronous sequential

designs that have a limited number of different flip-flops. In fully custom designs or in designs employing latches as opposed to flip-flops, the modeling methodology described here may not be suitable. Lastly, the models discussed here have been developed using a 2-micron process. Hence, these models also may not be suitable for deep submicron processes.

3.6 Actual Models Used

The above sections demonstrated that models can be developed using the more accurate SPICE level-two parameters. For the actual models used in the fault simulator, however, SPICE level-one parameters have been used. The reason is to allow a fair comparison between simulation results from TIFAS, ILLIADS [40] and SPICE. ILLIADS only supports SPICE level-one parameters, and SPICE takes considerably less time with level-one parameters compared to those for level-two.

The previous sections have shown a general modeling approach for the rise and fall delays. This approach can be used if there are a large number of different gate configurations. On the other hand, if the number of gates is limited, such as in a standard cell design, the propagation delays as a function of the output capacitance can be easily modeled by performing several SPICE simulations and putting the results in a table. This is the approach taken in TIFAS, and since the circuits used in these simulations are very small, the modeling process is not time intensive. When pin to pin delays are different due to the transistor topology, the average delay is used. For example, in a two-input

NAND gate, the falling transition times are different, dependent on which input is the last to rise high. In this case, the average of the two times is used.

The previous sections have also described a general modeling approach for the pulse widths. Although the model used is the same, different parameters have been derived for each gate to increase accuracy. However, only the pulse width at the point of charge injection has been explicitly modeled. Starting with the second-level gate, normal delay values are used to propagate the fault except that the first transition of the transient pulse is made to occur 0.1 ns faster at the output of the second-level gate to account for fast initial transition and overshoot of the pulse. It is shown in Chapter 5 that explicitly modeling only the injection node pulse width is still very accurate. It is possible to explicitly model the pulse width at the second-level gate as well if the need is identified in the future.

The flip-flops are also modeled with level-one SPICE parameters, and the latching windows obtained are very similar to those obtained with level-two parameters. *Dfnf311* has been modified to equalize the latching windows for both type-one and type-zero pulses because the original design is skewed in such a manner that the type-one pulse latching windows are larger and the type-zero pulse latching windows are almost nonexistent. Equalized latching windows provide for a more general circuit behavior under the presence of transient pulses, and these modified latching windows are shown in Table 3.3.

Lastly, alpha particle injected at a DFF input node is considered to be 0.1 ns wider as far as the DFF is concerned to account for the fast initial transition and the overshoot in the pulse, whose effect will be the same as that of a longer pulse.

Table 3.3 Modified dfnf311 latching windows (all numbers are in ns)

Pulse Duration	1-0-1 Pulse		0-1-0 Pulse	
	Earliest Time	Latest Time	Earliest Time	Latest Time
0.8	0.3	0.4	0.6	0.7
0.9	0.3	0.6	0.2	0.9
1.0	0.2	0.7	0.0	1.0
1.1	0.2	0.8	0.0	1.1
1.2	0.2	0.9	-0.1	1.2
1.3	0.2	1.0	-0.1	1.3
1.4	0.1	1.1	-0.1	1.4
1.5	0.1	1.2	-0.2	1.5

CHAPTER 4

TRANSIENT FAULT SIMULATOR

In this work, the gate level was chosen as the level of abstraction for simulation because it offers a reasonably high level of abstraction without sacrificing too much in topological details. As seen in Figure 1.1, the transient fault simulator FAST is composed of two distinct simulators, TIFAS and TPROOFS. In this chapter, each of the simulators will be discussed.

The need for a simulator with a rise/fall delay model is obvious because an alpha-particle hit, or any other transient fault, initially has critical timing information associated with it. For example, the delay from the fault injection point to the flip-flops as well as the duration of the transient voltage pulse are critical information which must be used in the simulation. This is the main reason for the development of TIFAS as opposed to a zero-delay simulator. However, once a fault is latched into a flip-flop, there are no more transient pulses propagating through the circuit, obviating the need for any timing information. In this case, a zero-delay logic simulator would suffice, and TPROOFS is used for this purpose.

4.1 Preprocessing Phase

The circuits used in this work are from the ISCAS-89 [54] suite of sequential benchmark circuits, used primarily in the testing community. These circuits were chosen because they are available, relatively small, and generally accessible. Two important tasks must be performed before the actual simulation takes place. The first task is the translation from the ISCAS-89 bench format to the internal format usable by TIFAS and TPROOFS. The second is the calculation of delays. Because the topology is static, the delay calculation has to be done only once before the actual simulation; it is practical to do it in the preprocessing phase so that all of the delay values as well as the circuit topology can be written to a file to be read by TIFAS.

The rise and fall delay values were derived from SPICE simulation of the cells from the standard cell library from Mississippi State University distributed as part of the Octtools set by the University of California, Berkeley, as described in the previous chapter. This cell library was chosen because the detailed information of the cell topology at the transistor level is available, enabling the simulation and modeling of the cells. Accurate modeling of these cells is important for a fair comparison of TIFAS to ILLIADS and SPICE. In this work the rise and fall delay values depend only on the fanout capacitance. Other parasitic elements such as routing capacitances are ignored.

4.2 Logic Simulation

A fault simulator is basically a logic simulator with the capability of injecting faults. The role of TIFAS is to simulate the injection of a voltage pulse at a node in the circuit and to propagate the pulse to the flip-flops to see whether the pulse is latched or not. It is based on the two-pass algorithm of only scheduling real events for further evaluation [55] and is shown in Figure 4.1. The algorithm works as follows. In the first pass, for every event (n, v) pending at the current time where n is the node number and v is the value, the node is updated, and all the gates on the fanout list of the node are added to the activated list. In the second pass, the gates on the activated list are evaluated one at a time, and if the output of the gate differs from the previously scheduled value (*lsv*), an event is generated and scheduled in the event queue in the appropriate time slot in the future, accounting for the gate delay. The advantage of this algorithm is that a gate activated by more than one event occurring at the inputs is guaranteed to be scheduled only once.

According to [55], the one-pass algorithm shown in Figure 4.2 of evaluating the gate as soon as it is activated is more efficient. This algorithm works as follows. For every event (n, v) pending at the current time, the node is updated, and all of the gates on the fanout list are examined immediately. If the output of a gate on the fanout list is different from the previously scheduled value, first a check is performed to see if the last scheduled time (*lst*) is the same as the time at which the currently generated event is to be scheduled. If they are the same, the last scheduled event is cancelled to suppress the

```
Two_Pass_Event_Driven_Simulation_Algorithm() {
  for every event(n, v) pending at current time {
    value(n) = v
    for every gate j on the fanout list of n {
      add j to the activated list
    }
  }
  for every j on the activated list {
    v' = evaluate(j)
    if v' != lsv(j) then {
      schedule(j, v') for time t+delay(j)
      lsv(j) = v'
    }
  }
}
```

Figure 4.1 A standard two-pass algorithm for event-driven simulation

zero width spikes which would have occurred as an artifact of the simulation algorithm. Then, the newly generated event is scheduled.

This algorithm may be more efficient, but it presents a problem when the rise and fall delays are different. In the algorithm shown in Figure 4.2, the rise and fall delays are assumed to be the same. When they are different, the *lst(j)* and t' should reflect the different delay values, making the algorithm more complex. For simplicity's sake, the two-pass algorithm was chosen.

Even with the two-pass algorithm, care must be taken when the rise and fall delays are different. Consider Figure 4.3. The inverter here has a rise delay of 10 and a fall delay of 5. At the input, there is a 1-0-1 pulse of width 2 occurring at time 0. At time 0, the inverter evaluates, and a 1 is scheduled for the output node at time 10. At time 2,

```
One_Pass_Event_Driven_Simulation_Algorithm() {
  for every event(n, v) pending at current time {
    value(n) = v
    for every gate j on the fanout list of n {
      v' = evaluate(j}
      if v' != lsv(j) then {
        t' = current_time + delay(j)
        if t' = lst(j)
          then cancel event (j, lsv(j)) at time lst(j)
        schedule (j, v') for time t'
        lsv(j) = v'
        lst(j) = t'
      }
    }
  }
}
```

Figure 4.2 A standard one-pass algorithm for event-driven simulation with zero-width spike suppression.

the inverter is evaluated again, and an event is scheduled at time 7. Left untreated, this simulator will erroneously report that both the input and the output of the inverter are 1 at time 10. In a case such as this, the second event generated at time 2 should override the first, in effect cancelling both events. This may be accomplished with the algorithm shown in Figure 4.4, which should be used immediately following the gate evaluations in the second pass of the two-pass algorithm. It works as follows. First, the newly evaluated value is checked for difference from the last scheduled value. If they are the same, then the new event is ignored. If they are different, the time t' at which the new event is to be scheduled is compared to the last scheduled time of the last event on this node. If t' is greater, it is scheduled. If not, the last scheduled event is removed from the queue, and the *lsv(j)* and *lst(j)* values are reset.

The logic system used in a simulator is one of its most important characterizing traits since the system can affect the efficiency as well as the capability of the simulator to a large degree. Generally, the more values there are in the logic system, the less efficient the simulator, but the greater the capability and the flexibility. Originally, TIFAS started out as a hierarchical simulator which could process VHDL descriptions. That is the main reason for using the *std_logic_1164* multivalue logic system, which is the standard logic system for VHDL. The logic system is shown in Table 4.1. This logic system can handle tri-state terminated outputs, as well as wired-OR type topologies.

The VHDL standard *std_logic_1164* also specifies truth tables to be used with the logic system. Two such tables are shown in Tables 4.2 and 4.3. Table 4.2 is the table showing

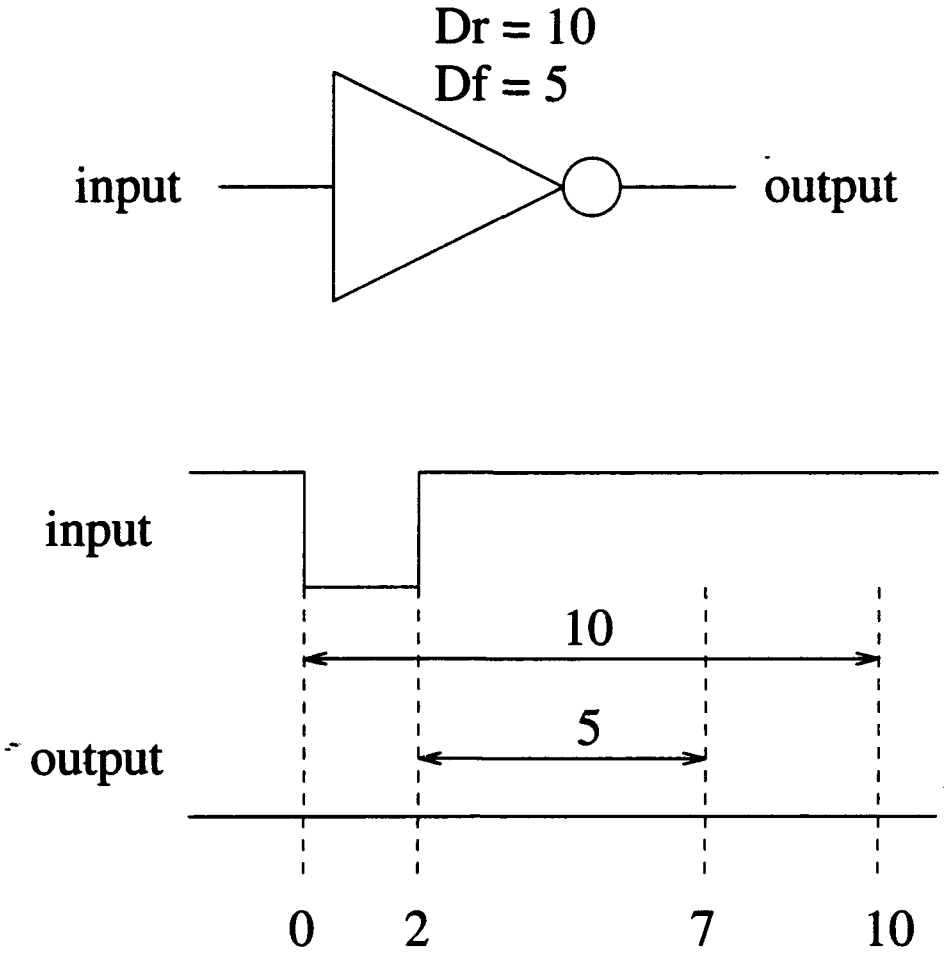


Figure 4.3 Cancellation of event due to different rise and fall delays

Table 4.1 The logic value system used in TIFAS

Value	Meaning
U	Uninitialized
X	Forcing Unknown
0	Forcing 0
1	Forcing 1
Z	High Impedance
W	Weak Unknown
L	Weak 0
H	Weak 1
D	Don't Care

```
Post_Evaluation_of_Gates() {
  if (v' = lsv(j)) {
    t' = current_time + delay for transition from lsv(j) to v'
    if t' > lst(j) {
      schedule(j, v') for time t'
      lst(j) = t'
      lsv(j) = v'
    }
    else {
      cancel event (j, lsv(j)) at time lst(j)
      update lst(j) and lsv(j)
    }
  }
}
```

Figure 4.4 An algorithm for processing the evaluation of gates with the rise/fall delay model to enable cancellation of previous events

what the value should be if two output lines are wired together, and is appropriately called the resolution function. Table 4.3 shows the truth table for an AND function.

The last major piece needed in a logic simulator is the timing wheel. Some sort of data structure is needed to maintain the header list which keeps track of the time and the events to be simulated at that time. The header list may be implemented as a linked list or as a fixed size array. Because of the inefficiencies of searching and updating a linked list, using an array as shown in Figure 4.5 is the preferred method. The figure shows a timing wheel of size M, which can keep track of events occurring between the current time T and T+M-1. For events occurring outside this range, an auxiliary linked list can be used to store them until they can be included in the timing wheel. However,

Table 4.2 VHDL standard resolution table

	U	X	0	1	Z	W	L	H	D
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	X
1	U	X	X	1	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
D	U	X	X	X	X	X	X	X	X

Table 4.3 VHDL standard AND table

	U	X	0	1	Z	W	L	H	D
U	U	U	0	U	U	U	0	U	U
X	U	X	0	X	X	X	0	X	X
0	0	0	0	0	0	0	0	0	0
1	U	X	0	1	X	X	0	1	X
Z	U	X	0	X	X	X	0	X	X
W	U	X	0	X	X	X	0	X	X
L	0	0	0	0	0	0	0	0	0
H	U	X	0	1	X	X	0	1	X
D	U	X	0	X	X	X	0	X	X

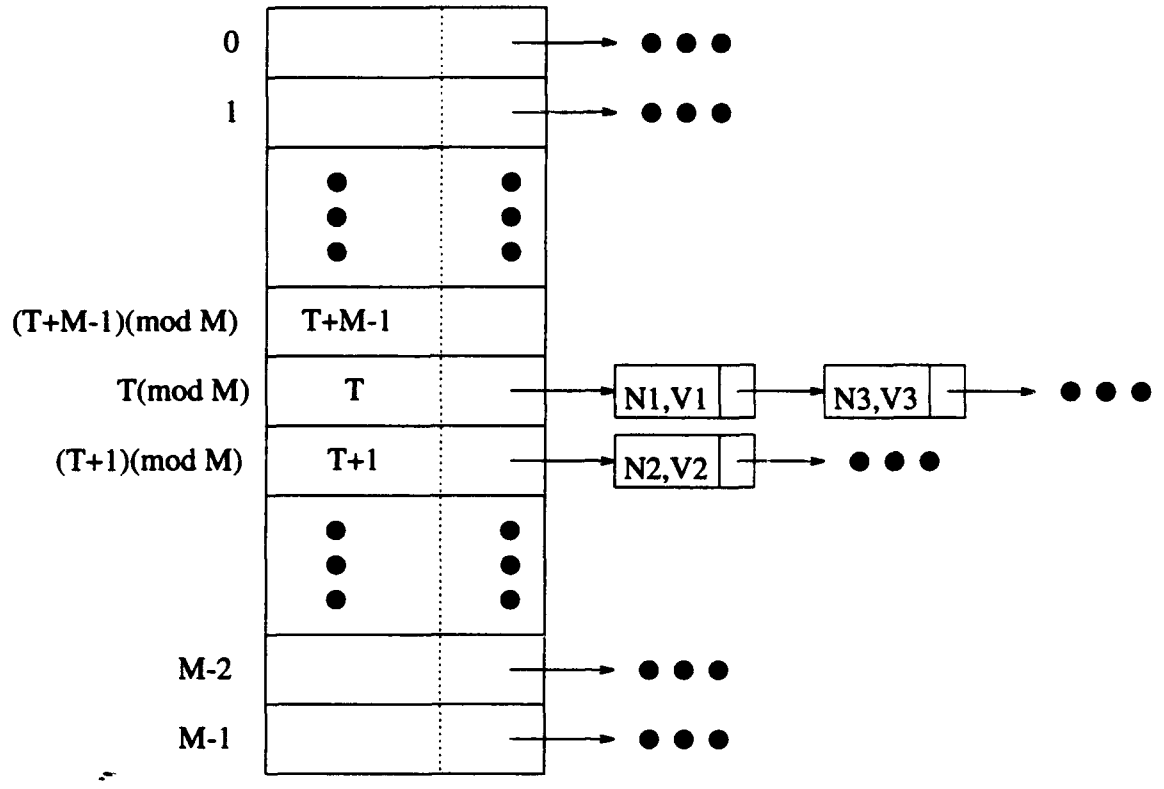


Figure 4.5 Timing wheel

because the next event generated by the current one usually occurs in the near future, with a reasonably large M, there is no need for the auxiliary linked list.

The wheel itself works as follows. The entries in the wheel correspond to the simulation time modulus M. Therefore, the entry T (modulus M) contains events for time T, and the entry T+1 (modulus M) contains events for time T+1. The entry immediately preceding T, instead of containing events for T-1, contains events for the time T+M-1. Indeed, the name timing wheel comes from the circular behavior of the array arising from the modulus operator. In TIFAS, the basic unit of time was chosen to be 0.1 ns, and the timing wheel size was chosen to be 100. Even with the timing wheel of this size, there was no need for the auxiliary linked list.

4.3 Timing Fault Simulator

The simulation of a transient fault differs from stuck-at fault simulation in that timing is involved. In stuck-at fault simulation, various techniques are used to speed up the simulation time, such as concurrent, deductive, differential, and parallel simulation algorithms [14]. For transient fault simulation, however, these techniques are not applicable because of the nature of the transient fault. A transient fault, defined here as a random voltage pulse occurring at a node resulting from some outside source, inherently has timing information which renders the above techniques ineffective or useless. For that reason, the serial fault simulation technique was chosen.

An alpha-particle-induced transient fault is specified by the injection node, the injection time and the total injected charge. Since transient faults occur randomly, all three parameters are random. It follows that the domain of the transient faults is very large, necessitating a fast transient fault simulator.

If the standard algorithm for event-driven simulation is employed to perform the fault simulation at the gate level, all the events that occur in the clock cycle of fault injection must be simulated. These events include the transient pulse as well as the input and the DFF events. This algorithm is illustrated in Figure 4.6. Normally, many transient faults are simulated in a given clock cycle, and if we can possibly avoid re-simulating the input and the DFF events every single time, we will obtain a significant speedup in the simulator.

```
Standard_Event-Driven_Algorithm() {
  For each input vector {
    Schedule input vector and DFF outputs in timing wheel
    Evaluate good circuit
    Save good circuit state
    While transient fault remains to be injected in this cycle {
      Initialize faulty circuit state to be the good circuit
          state from the previous cycle
      Schedule input vector, DFF outputs and transient fault
          events in timing wheel
      Evaluate
      Compare DFFs with the good circuit
    }
  }
}
```

Figure 4.6 A standard event-driven algorithm for transient fault simulation

The transient pulses of interest to us are those which can be latched into a flip-flop. Since the latching window is near the clock edge and pulses have to arrive inside the latching window to be latched into flip-flops, it seems reasonable to assume that the interesting pulses propagate through the circuit after the normal events have passed through. If that is indeed the case, we only have to propagate the transient pulse with the other nodes set to the stable values for the clock cycle. This would provide large savings in simulation time since the normal events for the clock cycle need not be re-simulated every time a fault is injected. Figure 4.7 illustrates this algorithm.

The fault-driven algorithm produces the same simulation output as the standard event-driven algorithm under certain conditions as formalized in the following assertion:

```
Fault-Driven_Algorithm() {
  For each input vector {
    Schedule input vector and DFF outputs in timing wheel
    Evaluate good circuit
    While transient fault remains to be injected in this cycle {
      Schedule transient fault events in timing wheel
      Evaluate
      Compare DFFs with the good circuit
    }
  }
}
```

Figure 4.7 A fault-driven algorithm for performing transient fault simulation

Assertion 1. *For synchronous circuits composed of simple gates with transport delay satisfying the conditions that*

1. *the latest possible transition at the inputs to latches occurs at time*

$$t_f = t_{clk\_edge} - setup\_time - pulse\_width \quad and$$

2. *the transient pulse does not reconverge on itself,*

only the faulty transitions need to be evaluated with the other nodes holding the final stable values for the clock cycle, irrespective of the fault injection node and time.

Proof: There are two cases to consider: (1) the second transition of the transient pulse arrives at the input of a flip-flop at time $t \leq t_{clk\_edge} - setup\_time$, and (2) the second transition of the transient pulse arrives at time $t > t_{clk\_edge} - setup\_time$. If the transient pulse reconverges on itself, it is possible to have a combination of cases (1) and (2) where the pulse width at a flip-flop input may be different for the two algorithms discussed

above, which may result in different outputs. However, the second condition in the assertion requires that the pulse does not reconverge on itself.

For case (1), the pulse arrives at such a time that it will not be latched by the flip-flop in the fault-driven simulator. Consider what happens in the standard event-driven simulator. Since the normal transitions at the input of the flip-flop die out before the time $t_{clk\_edge} - setup\_time$, the normal transitions cannot cause the second edge of the transient pulse to occur after time $t_{clk\_edge} - setup\_time$. Therefore, this pulse will not be latched in the standard event-driven simulator either.

For case (2), the pulse arrives at a time when it may be latched depending on its duration. Consider what happens to the first edge of the transient pulse in the standard event-driven algorithm. The first edge arrives at the input to the flip-flop at time $t > t_{clk\_edge} - setup\_time - pulse\_width$. Because of the first condition in the assertion, this edge arrives after the arrival of the last normal transition. In fact, at every node along the path from the injection node to the flip-flop, the first edge will have occurred after the arrival of the last normal transition to that node in order to satisfy the same condition. Therefore, the first edge may be evaluated assuming all nodes have settled to their stable values for the clock cycle. The second edge may be evaluated in this manner as well since it occurs after the first edge. Therefore, the fault-driven algorithm and the standard event-driven algorithm will produce identical outputs for this case. □

Both algorithms have been implemented and experiments have been conducted to see the difference in the outputs of the two algorithms in realistic situations where both of the conditions in the assertion may be violated. Table 4.4 shows the experimental

results on ten circuits from the ISCAS-89 suite of benchmark circuits. At the time of this experiment, the fault simulator could only support a transport delay; furthermore, the delay models were not yet developed, so reasonable values for delay from LSI Logic's standard cell library [56] were used. Also, the pulse-width model was not ready at the time, so logic pulses randomly chosen from 0.7 to 1.5 ns in duration were injected. Finally, the DFF latching window is the original latching window, with the flip-to-zero window made the same as the flip-to-one window. The clock periods in the simulations were set to minimally satisfy the setup time requirements of the DFFs. In each of the circuits, 100,000 randomly chosen transient pulses were injected. The input vector files used were generated by an automatic test pattern generation tool [57]. Figure 4.8 summarizes the experimental setup. The column **Latched Pulses** shows the number of transient pulses which were latched into flip-flops, and the column **Difference** shows the number of transient pulses which resulted in different outcomes from the two algorithms. As the table shows, the differences between the two algorithms are negligible; furthermore, most of the differences were compensating. That is, for every fault latched in only one simulator, there tended to be another fault which was only latched in the other simulator. Therefore, we can safely use the fault-driven algorithm.

The time spent in execution was measured in the experiment conducted above to see the speedup obtained in the fault-driven algorithm compared to the standard one. The results are shown in Table 4.5. As the table shows, the fault-driven algorithm achieves a speedup of up to 36 compared to the standard one.

Table 4.4 Comparison of the outputs of the two algorithms

Circuit	Latched Faults	Difference
s208	4,839	0
s641	3,113	3
s713	3,067	4
s820	1,995	19
s953	1,146	0
s1196	1,466	0
s1238	1,493	0
s1494	1,013	3
s5378	2,685	0
s35932	2,896	10

```
Number of faults injected:     100,000 per circuit
Fault injection node:          randomly chosen over all nodes
Transient pulse width:         randomly chosen from 0.7 ns to 1.5 ns
Fault injection time:          randomly chosen over the whole clock period
Fault injection clock cycle:   evenly distributed over the middle
                                 portion of the simulation run
Input vector suite:            automatically generated from
                                 Sequential Circuit Test Generator (STG)
Clock period:                  chosen to minimally satisfy DFF setup time.
```

Figure 4.8 The experimental setup for validating the improved algorithm

Table 4.5 Comparison of the speed of the algorithms with 100,000 fault injections

Circuit	Gates	Standard Alg. (min)	Fault-Driven Alg. (min)	Speedup
s208	104	6.2	2.2	2.8
s641	398	17.8	4.7	3.8
s713	412	20.5	4.9	4.2
s820	294	20.4	2.9	7.0
s953	424	19.0	1.4	13.6
s1196	547	29.6	3.9	7.6
s1238	526	30.6	4.0	7.7
s1494	653	46.8	4.2	11.1
s5378	2,958	99.7	8.3	12.0
s35932	17,793	1567.2	43.3	36.2

Figure 4.9 shows the speedup as a function of the gate count. Although the data are insufficient to make a definite statement about the speedup improvement as a function of the gate count, the graph does indicate a larger speedup with the increase in the number of gates. This trend is important and argues in favor of the fault-driven algorithm since the simulator is targeted for large circuits.

As a further verification that the speedup is due to algorithmic rather than coding efficiency, the number of events generated was also counted in the above experiment. Table 4.6 shows the comparison of the two algorithms in terms of the number of events. The table shows that the number of events is drastically reduced in the fault-driven algorithm compared to the standard one. Figure 4.10 shows the reduction factor in the number of evaluations as a function of gate count. Similar trend can be seen here as in Figure 4.9.

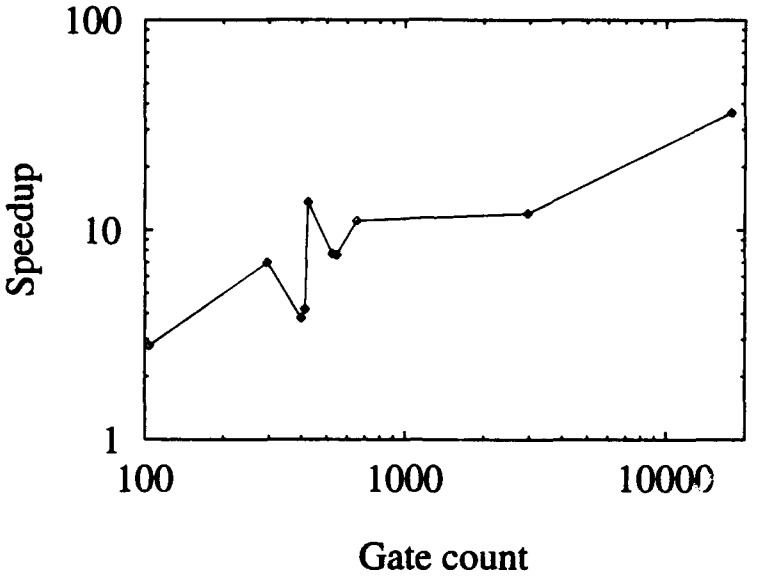


Figure 4.9 Speedup as a function of gate count

Table 4.6 Comparison of number of events (in millions)

Circuit	Stand.	F-D	Reduc. Factor
s208	4.0	0.7	5.7
s641	15.8	1.6	9.9
s713	19.2	1.7	11.3
s820	14.1	0.7	20.1
s953	13.7	0.5	27.4
s1196	24.9	0.9	27.7
s1238	23.9	0.8	29.9
s1494	42.4	1.1	38.5
s5378	108.1	1.7	63.6
s35932	1,613.0	2.1	768.1

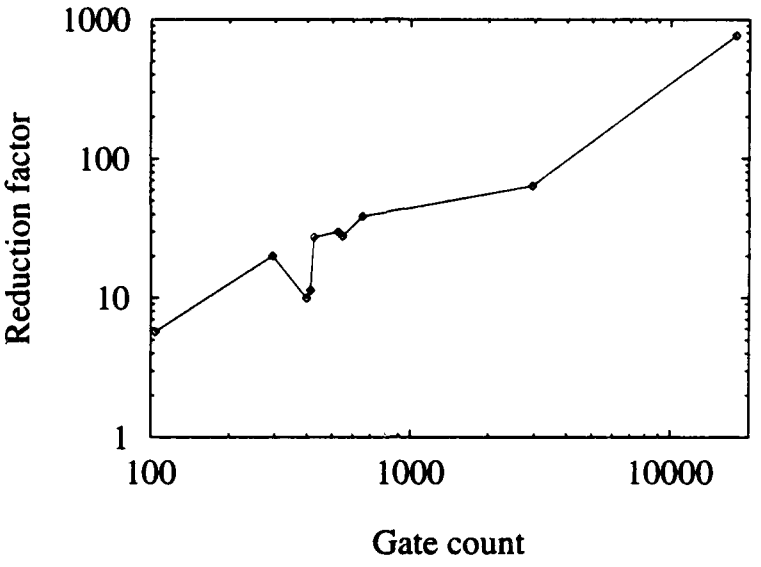


Figure 4.10 Reduction factor as a function of gate count

4.4 TPROOFS

Once the errors are latched, TIFAS is no longer needed since the circuit can now be simulated with full accuracy using a more efficient zero-delay logic simulator. PROOFS [14] is a fast and memory efficient zero-delay parallel fault simulator which can be used for this purpose.

Transient PROOFS (TPROOFS) is a modified version of PROOFS which allows faults to be injected at the outputs of DFFs. The latched errors in TIFAS are collected and written to a file, which is then read by TPROOFS to carry out the fault simulation for the remaining vectors in the test sequence. In TPROOFS, any number of faults can be simulated in a particular clock cycle, and any number of DFF flips for a given fault can be correctly simulated. The algorithm used in TPROOFS is shown in Figure 4.11.

```
TPROOFS-Algorithm() {
  For each input vector {
    Evaluate good circuit
    Save good circuit state
    While faulty machines remain {
      Get next group of 32 active transient faults
      Set faulty machine flip-flops for faults which occurred in the
          previous time frame or whose fault effects propagated to
          flip-flops in the previous time frame
      Evaluate faulty machines
      Save faulty machine flip-flop values
    }
  }
}
```

Figure 4.11 The algorithm used in TPROOFS, a zero-delay parallel fault simulator

TPROOFS is particularly fast because it is a zero-delay parallel simulator. The zero-delay circuit model allows it to use efficient data structures, and combined with the parallel simulation of 32 active faults, TPROOFS can achieve from one to two orders of magnitude speedup over serial or even concurrent fault simulators.

CHAPTER 5

ACCURACY OF THE TIMING FAULT pSIMULATOR

The validation of the transient fault simulator described above is made difficult for two reasons. The first problem is the inherent difference between a logic level simulator and an electrical level simulator. This difference invariably results in different delays for signal propagations through the circuit. Therefore, even if the latching behaviors of the DFFs are modeled as accurately as possible, the two simulators may report different results on the same injected fault due to the different arrival times of the injected pulse.

The second problem is the long simulation times incurred when running SPICE. In fact, the time cost of SPICE is one of the primary reasons for developing FAST. This problem places limits on the validation process: one, large circuits cannot be simulated, and two, even small circuits cannot be simulated with a large number of fault injections. For example, SPICE3 takes about two minutes per injection for the circuit s208 on a SUN ELC workstation. Simulation of 100,000 fault injections would take more than four months at this rate.

These problems have forced the validation experiments to be scaled back. For the validation experiment, only five circuits are used. Among these, it is feasible to simulate a relatively large number of fault injections in SPICE on only the smallest circuits s27 and s208. For the other circuits, s641, s713, and s820, another technique is used.

ILLIADS [40] is a switch-level timing simulator which achieves faster simulation times compared to SPICE with little loss in accuracy by using the analytic solution of the Ricatti equation. It has been modified to support current injection to simulate alpha-particle hits [16]. Because this is an accurate timing simulator, the results from this simulator should be relatively close to the results from SPICE. For these reasons, ILLIADS is primarily used to validate TIFAS for the circuits s641, s713 and s820.

5.1 Comparison to SPICE and ILLIADS

SPICE3 was originally used for verification purposes. However, it reported unreliable simulation results. For example, with the same circuit description and the same injected fault in the same clock cycle, SPICE3 reported different latching behaviors depending on the length of the transient simulation, although the results should have been the same. Moreover, SPICE3 has convergence problems with some faults. For those reasons, HSPICE has been used instead.

The DFF used here is a modified version of the original DFF where the latching windows have been equalized. This was done to observe a more general circuit behavior, rather than observing only unidirectional DFF flips.

A word should be said about the relative speed of the simulators. Table 5.1 shows the simulation times. From the table, we can see that ILLIADS is from 10 to 30 times faster than HSPICE, whereas TIFAS in turn is from three to four orders of magnitude faster than ILLIADS.

Table 5.1 Simulation times for SPICE, TIFAS, and FAST

Circuit	Gates	SPICE (sec/fault)	ILLIADS (sec/fault)	TIFAS (sec/10,000 faults)
s27	13	44.0	4.6	12.1
s208	104	304.4	10.2	20.0
s641	398	1298.6	41.4	46.3
s713	412	1371.6	45.9	46.4
s820	294	1395.7	44.4	37.6

For this experiment, 2,000 random faults have been injected in each of the five circuits. The faults were chosen randomly in space and time, and evenly distributed from 1 pC to 9 pC. The only nodes not injected with faults are primary inputs, primary outputs, and DFF outputs. The DFF output nodes were excluded because an alpha-particle hit on the DFF output node may cause the DFF to flip its state, which is a different phenomenon from the one of interest here.

Table 5.2 shows the results of this experiment. The second, third and fourth columns show the number of latched faults reported by each simulator. For the circuits s27 and s208, all 2000 fault injections were performed in TIFAS, ILLIADS and SPICE. For the circuits s641, s713 and s820, each fault injection simulation in SPICE takes about 30 min on a SUN Sparkstation 1 workstation, making it unfeasible to simulate all 2000 fault injections. Instead, TIFAS and ILLIADS simulations were done first, and the union

of the latched faults reported by TIFAS and ILLIADS were then simulated in SPICE. Columns 5 and 6 show the number of latched faults reported by SPICE that were also reported latched by TIFAS and ILLIADS, respectively.

Table 5.2 Comparison of latched faults between SPICE, TIFAS, and FAST

Circuit	Latched			Common with SPICE	
	SPICE	TIFAS	ILLIADS	TIFAS	ILLIADS
s27	116	128	146	85	109
s208	35	32	47	28	32
s641	4	12	12	3	2
s713	6	16	17	3	6
s820	1	5	6	1	1

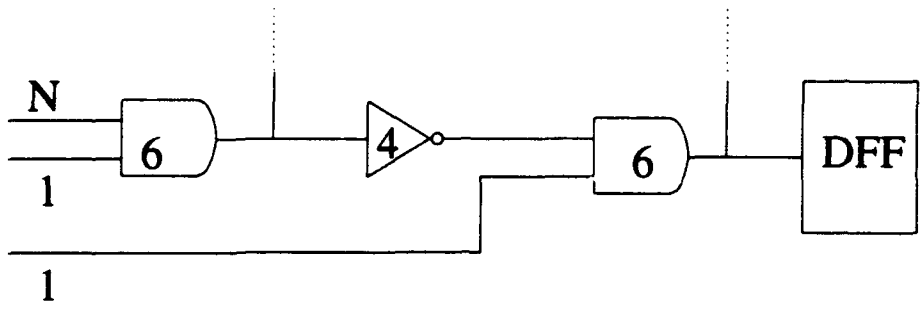
The number of faults reported latched is arguably the most important parameter since it indicates the sensitivity of the circuit to an alpha-particle hit. For the circuits s27 and s208, the numbers of faults reported latched by TIFAS are off by about 10% compared to the ones reported by SPICE. For the larger circuits, the numbers are not close. However, as mentioned above, since only the faults that were latched by TIFAS or ILLIADS were simulated in SPICE, the actual number of faults latched by SPICE would probably be higher if all 2,000 fault injections per circuit are simulated.

For the circuit s27, SPICE reported 116 latched faults out of 2,000 injections. Of these 116, 85 (73%) were also reported latched by TIFAS. For the circuit s208, 80% of the faults reported latched by SPICE were also reported latched by TIFAS. The percentage for the larger circuits cannot be determined since we do not have complete data on the number of faults latched in SPICE.

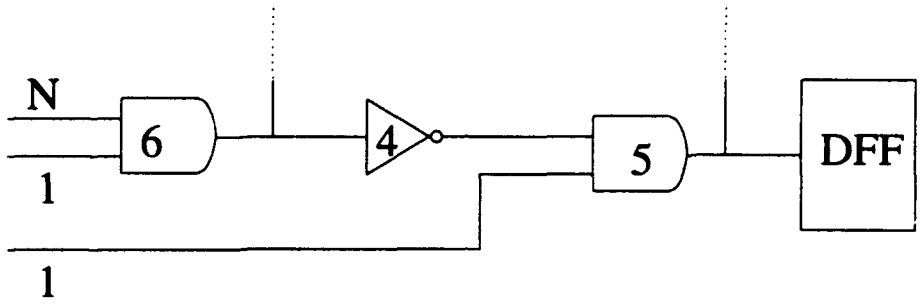
Some comments should be made about these results. Although TIFAS is not as accurate as SPICE, TIFAS is shown to be within 10% of SPICE for the small circuits as far as the number of faults latched is concerned. If the application does not call for SPICE-like accuracy as far as the number of faults latched is concerned, TIFAS can be used. Also, although TIFAS did not catch all of the latched faults reported by SPICE, it did catch more than 70% of the latched faults for the smaller circuits, at a small fraction of the computational cost of SPICE. This capability may be important in certain applications. For example, when designers are trying to make the circuit more transient fault tolerant, TIFAS can give quick feedback on at least some of the faults that would be reported latched by SPICE.

5.2 Main Cause Behind the Different Simulation Results

As mentioned in the beginning of this chapter, difference in delays is a major reason behind the different results reported by the three simulators. Let us consider the circuit shown in Figure 5.1. In (a) and (b) the circuit is shown with delay values used in TIFAS and ILLIADS, respectively. For simplicity, assume that the rise and fall delays are the same. Now suppose that a pulse is injected at node N 15 units before the earliest time of the latching window corresponding to the pulse duration. The fault will be latched in ILLIADS simulation, but it will not be in TIFAS. On the other hand, if a pulse is injected at node N 16 units before the latest time of the latching window, it will be latched in



(a) Delays in TIFAS



(b) Delays in ILLIADS

Figure 5.1 Same circuit with different delays in TIFAS and ILLIADS

TIFAS but not in ILLIADS. Thus, the difference in delay amounts to a translation in time of the latching window. Provided that the pulse durations and the latching windows are the same in both simulators, the percentage of randomly injected faults that are latched will be equal in the two simulators. Of course, the pulse widths are not guaranteed to be the same since different rise and fall times can attribute to either the widening or the narrowing of the pulse. However, the close correspondence between the latched faults reported by TIFAS and ILLIADS gives strong indication that the translation of latching windows is the major reason behind the difference in the simulation results.

CHAPTER 6

EXPERIMENTAL RESULTS ON BENCHMARK CIRCUITS

In this chapter, FAST is used to study the characteristics of benchmark circuits under alpha-particle hits. The circuits chosen for the experiment are from the ISCAS-89 suite of sequential benchmark circuits, as done in Chapter 5. The DFF used is the modified DFF whose equalized latching windows are shown in Table 3.3.

6.1 Experimental Conditions

The experimental conditions here are similar to the conditions used above. For each circuit, one-million faults were randomly chosen for injection. All output nodes of gates were possible candidates for injection, except primary output nodes and DFF output nodes. The clock period was set to be 2 ns longer than the latest settling time in the circuits. The total charge injected varied from 1 pC to 9 pC in 1 pC increments. Finally, the injection times were chosen randomly.

6.2 Experimental Results

Table 6.1 shows the number of faults that were latched into at least one DFF, as well as the distribution of the number of DFF flips for a given fault. For example, the column **Two** shows the percentage of latched faults which resulted in two DFF flips. The numbers show that overall, between 86 and 99 percent of all latched faults result in a single DFF flip. This number is important for the researchers injecting faults at the register level, since most fault injections performed at this level inject a fault by flipping a single bit. Depending on the circuit, and depending on the application, single-bit flip may be an acceptable model. For example, if one is looking at the gross behavior of a system under faults, a single-bit flip model may suffice. However, a significant percentage of faults did result in two or more DFF flips that for validating highly fault-tolerant systems, it would be better to use TIFAS to derive a register-level model which includes multiple DFF flips or to use TIFAS to find the specific DFFs that were flipped.

Table 6.1 Latch flip distributions with one-million injections per circuit

Circuit	Latched Faults	DFFs Flipped (percentage)				
		One	Two	Three	Four	Five or More
s27	68,219	91.2	8.8	0.0	0.0	0.0
s208	14,663	95.6	3.4	0.9	0.1	0.0
s641	8,874	90.6	8.7	0.5	0.2	0.0
s713	9,979	94.0	5.3	0.6	0.2	0.0
s820	6,409	99.6	0.4	0.1	0.0	0.0
s1196	5,474	97.1	2.6	0.2	0.0	0.0
s1238	5,305	97.1	2.5	0.3	0.1	0.0
s1494	1,030	96.2	3.2	0.6	0.0	0.0
s5378	11,147	86.4	6.5	2.9	2.0	2.3
s35932	9,653	99.0	0.0	0.0	0.0	0.9

Table 6.2 Data on ISCAS-89 circuits

Circuit	Gates	Potential Injection Sites	DFFs	Clock Period (ns)
s27	13	9	3	5.9
s208	104	94	8	8.7
s641	398	355	19	13.8
s713	412	370	19	14.0
s820	294	270	5	13.8
s1196	547	515	18	14.1
s1238	526	494	18	15.2
s1494	653	628	6	19.3
s5378	2,958	2,730	179	14.2
s35932	17,793	15,745	1,728	14.5

To further analyze Table 6.1, we need other data on the circuits shown in Table 6.2. Part of the reason why the smaller circuits don't register more than five flipped DFFs is that there are only a few DFFs to begin with, as can be seen in the table. For the largest circuits s5378 and s35932 with a large number of DFFs, there are more faults resulting in a higher number of flipped DFFs.

A curious phenomenon was observed when the DFF flip distributions were analyzed with respect to the total injected charge. Figure 6.1 shows the DFF flip distributions for various amounts of injected charge for the circuit s5378. As can be seen in the figure, the total injected charge seems to have little effect on the distribution, especially above 2 to 3 pC of injected charge. The same general trend is observed in other circuits as well, and Figure 6.2 shows the distribution for the circuit s1196. These data indicate that small errors in the pulse width model will not have a major impact on the final DFF flip distributions. As both figures show, the number of faults causing multiple DFF flips tails off exponentially.

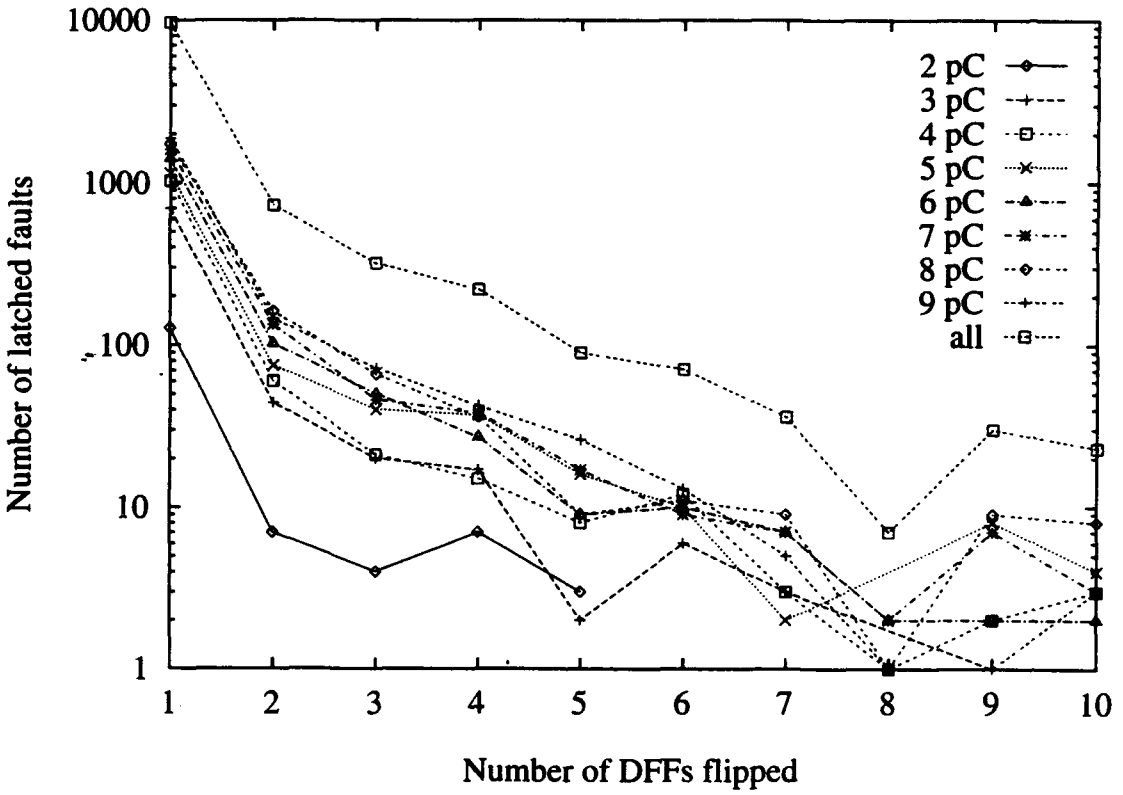


Figure 6.1 The DFF flip distributions for various amounts of total injected charge for the circuit s5378

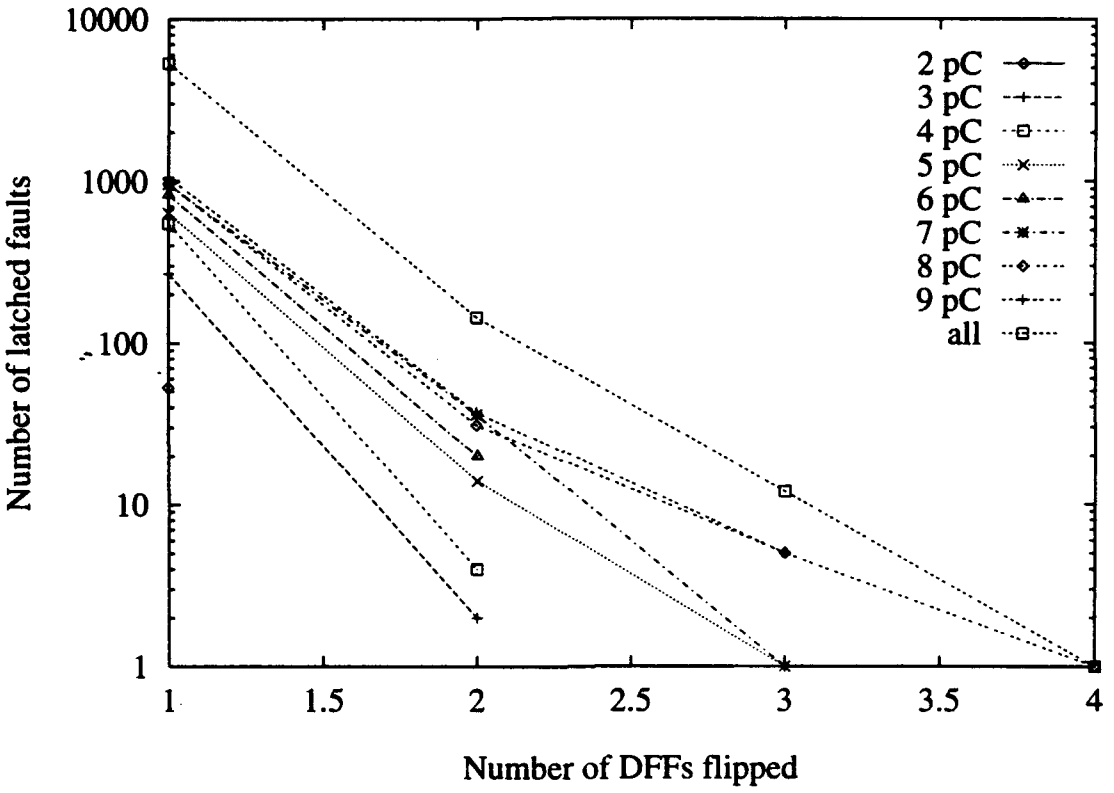


Figure 6.2 The DFF flip distributions for various amounts of total injected charge for the circuit s1196

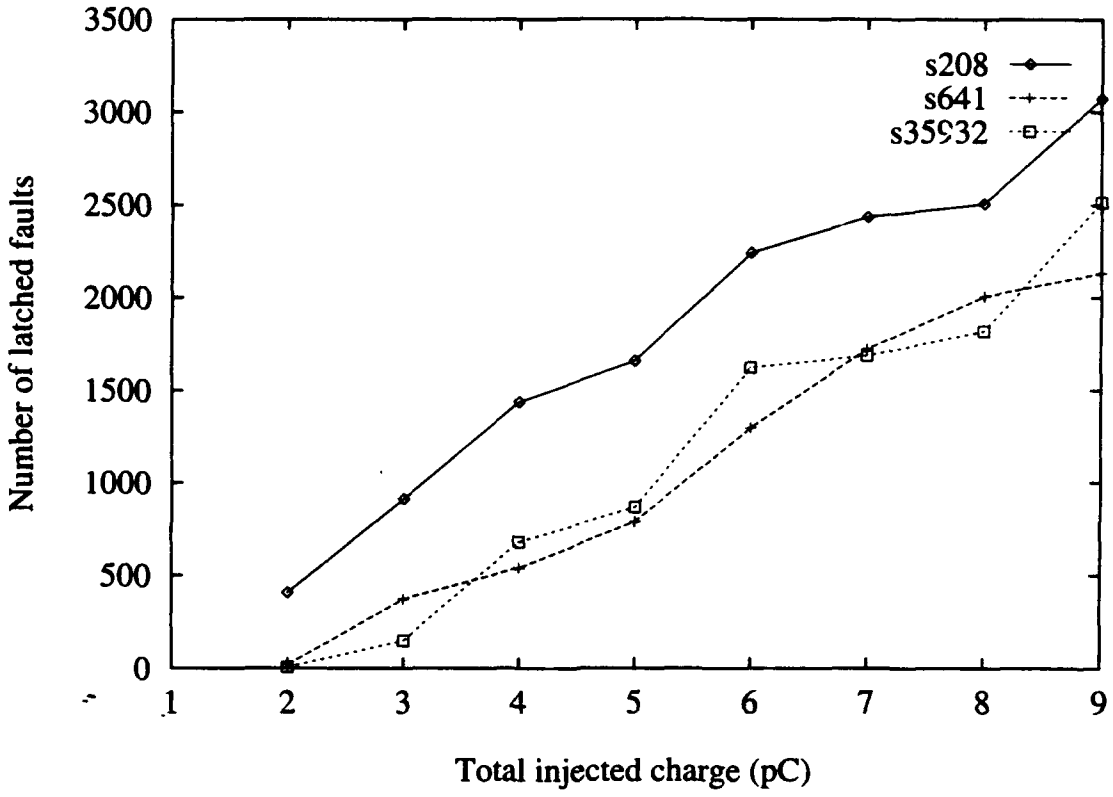


Figure 6.3 The number of faults latched as a function of the total injected charge for the circuits s208, s641 and s35932

Figure 6.3 shows the number of faults that were latched as a function of the injected charge for the circuits s208, s641 and s35932. As expected, the higher the charge, the greater the likelihood the fault will be latched for all three circuits. This is a general trend which holds for the other circuits as well.

Figure 6.4 shows the distribution of the faults that were latched with respect to their injection times. The x-axis denotes the number of seconds that the fault was injected prior to the clock edge, and the y-axis denotes the number of faults that were latched for the specific time of injection. Different circuits showed different behavior here. For the

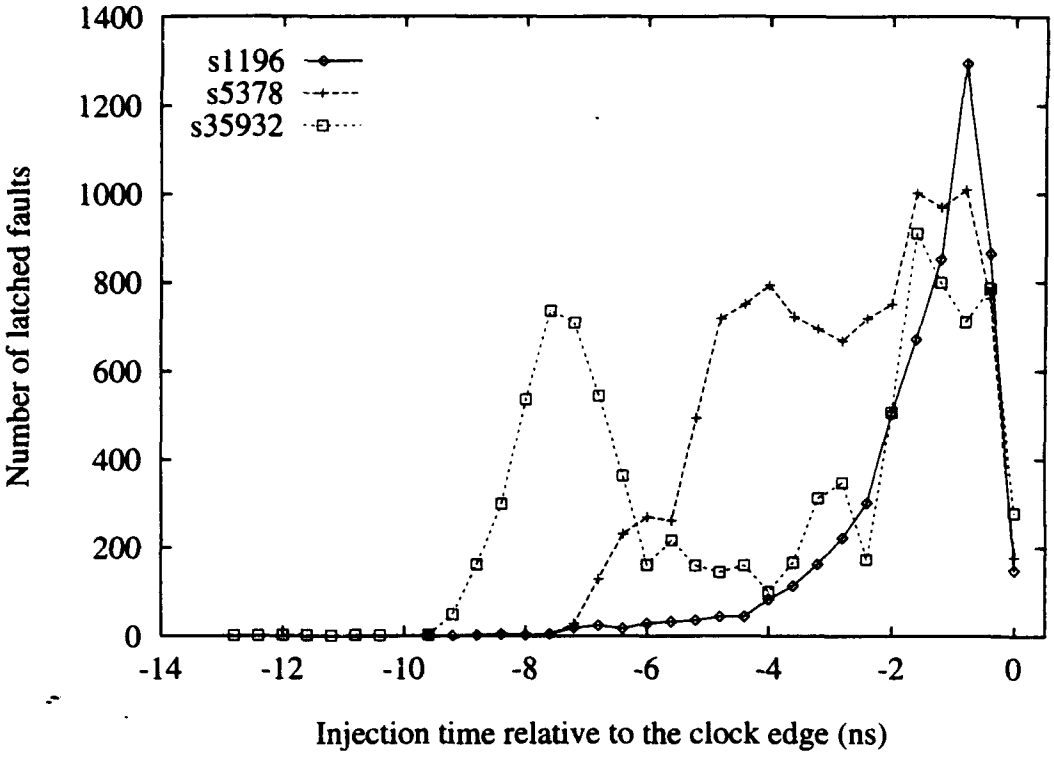


Figure 6.4 The distribution of the number of latched faults with respect to various injection times for the circuits s1196, s35932 and s5378. Clock edge occurs at time 0.

circuit s1196, the distribution is concentrated around 1 ns before the clock edge, whereas circuit s5378 shows a more rounded distribution in the range 0 to 5 ns prior to the clock edge. Circuit s35932 shows two peaks, one near the clock edge and one about 7 to 8 ns away.

Still, most of the circuits show a peak near the clock edge which means that the faults that were injected closer to DFFs have a higher probability of being latched. This is reasonable since faults injected farther away from DFFs have a higher probability of being logically blocked. However, the graph for circuit s35932 in Figure 6.4 shows that this generalization cannot be made for all circuits.

There may be certain nodes in the circuit which may be more sensitive to alpha-particle hits than others. From the previous observation on the injection time, it would seem that the nodes closer to the DFFs would be more sensitive. Table 6.3 shows the number of nodes that account for at least 50% of all latched faults. In the larger circuits, less than 10% of the nodes accounted for more than 50 percent of latched faults. If the number of sensitive nodes is small, it may be possible to redesign the circuit to make the nodes less sensitive, but this approach probably is not very attractive for most large circuits as there are a large number of sensitive nodes.

Table 6.3 also shows the number of DFFs which accounts for 50% of the total flipped bits. If the sensitive DFFs are not in the critical path of the circuit, it may be possible to use the pulse tolerant DFF design presented in the next chapter to increase the transient fault tolerance of the circuit with little or no performance penalty.

Finally, Table 6.3 also shows the number of flip-to-0 and flip-to-1 DFF flips. In most circuits they are evenly balanced, but in circuits s820 and s5378 there are more flip-to-1 flips by a ratio of about 4 to 1. In these cases, the DFF design may be modified to have smaller flip-to-1 latching windows at the expense of larger flip-to-0 latching windows to obtain fewer latched faults overall.

The latched errors were run through TPROOFS, and the transient faults causing errors on the output pins as well as the latency data were gathered. In this thesis, the transient faults showing up at the outputs in TIFAS were not considered as errors propagating to output pins: only latched errors propagating to the output pins in TPROOFS were considered as errors at the output. The input vector [57] was lengthened by 50

Table 6.3 Most sensitive nodes and DFFs

Circuit	Sensitive Nodes	% of Total Nodes	Sensitive DFFs	% of Total DFFs	Flip-to-0	Flip-to-1
s27	3	33	2	33	32,437	41,757
s208	7	7	4	50	4,964	10,498
s641	57	16	6	32	4,597	5,190
s713	60	16	6	32	3,651	7,033
s820	24	9	2	40	1,289	5,155
s1196	31	6	6	33	2,757	2,888
s1238	32	6	6	33	2,525	2,957
s1494	41	6	3	50	392	683
s5378	258	9	22	12	3,650	11,030
s35932	1896	12	188	11	3,244	7,800

cycles to give the latched faults more time to propagate to the primary outputs. The TPROOFS simulation was very fast, taking only 618 sec to simulate all ten circuits with 140,753 faults on a Sun Sparkstation ELC. Table 6.4 shows the latched faults which propagated to the primary outputs. As we can see, the percentage of the latched errors which propagated to the primary outputs varies widely from circuit to circuit, from a minimum of 32% to a maximum of 98%.

The error manifestation latency is defined as the first cycle in which the latched fault propagated to at least one output minus the cycle in which it was injected in TPROOFS. Figure 6.5 shows the error manifestation latency for the circuits s208, s713 and s5378. The latency distributions are not the same for different circuits, but the general trend is the same: the latched faults tend to propagate to the primary outputs sooner rather than later.

Another interesting aspect to examine is the pin error distribution which is important to the researchers injecting faults at the pin level. Since TPROOFS reports all the

Table 6.4 Latched faults propagating to primary outputs (one-million injections per circuit)

Circuit	Latched Faults	Primary Output Errors	Percentage
s27	68,219	32,511	48
s208	14,663	4,742	32
s641	8,874	4,742	48
s713	9,979	4,578	46
s820	6,409	5,516	86
s1196	5,474	1,743	32
s1238	5,305	1,667	31
s1494	1,030	1,006	98
s5378	11,147	4,069	37
s35932	9,653	5956	62

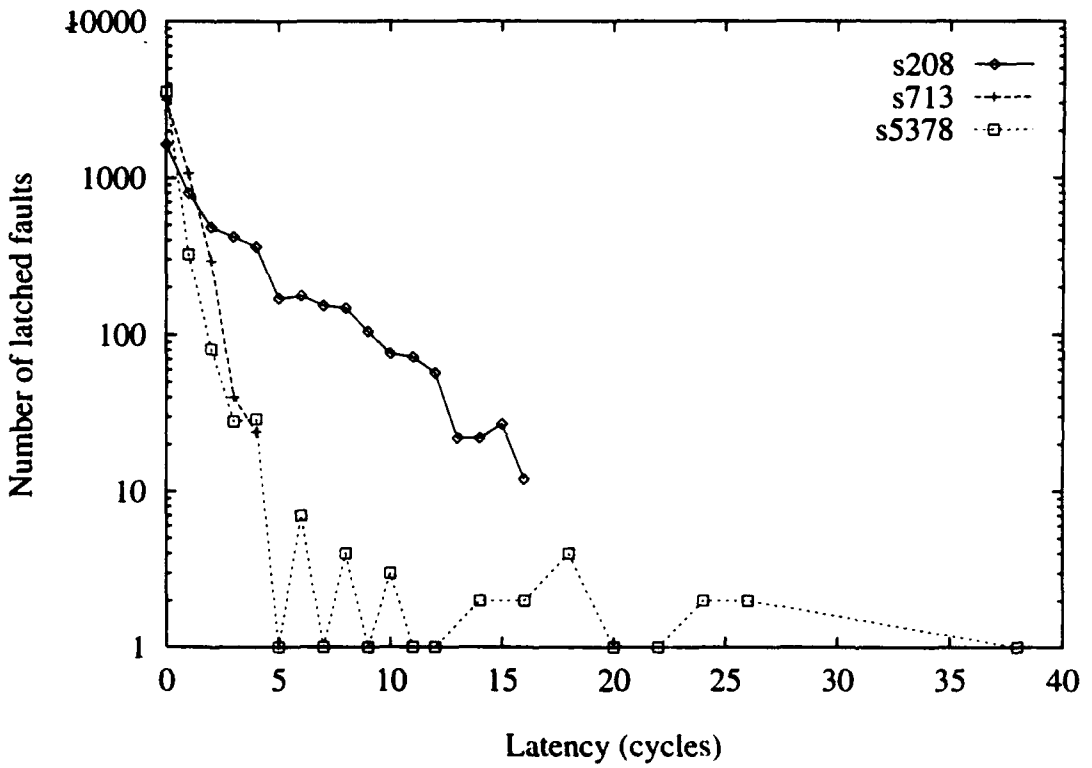


Figure 6.5 The latency of the latched faults propagating to the primary outputs for the circuits s208, s713 and s5378

primary outputs that are affected by latched faults, the pin error distribution can be obtained by post-processing the TPROOFS output. Table 6.5 shows the distributions for the circuit s713. The manifestation cycles are defined with respect to the first cycle in which the fault manifested itself at the pins. For example, manifestation cycle of 0 refers to the first cycle in which the fault propagated to the pins, 1 refers to the next cycle, etc. Other circuits displayed different distributions. For example, for the circuit s1196, all the fault effects disappeared after the third cycle of manifestation, whereas for the circuit s1494, the fault effects lingered for more than 20 cycles after the first cycle of manifestation. For the circuits in which the fault effects last for more than one cycle, the pin error distributions do not stay constant, nor do they decrease in a predictable manner. Therefore, there is no easy way to model latched faults at the pin level for the circuits in general.

Table 6.5 Pin error distributions according to the clock cycle of manifestation for the circuit s713

Manifestation Cycle	Number of erroneous pins			
	1	2	3	4
0	4,030	449	63	36
1	1,345	445	26	8
2	652	382	23	14
3	350	167	25	0
4	132	71	3	0
5	42	7	0	0
6	5	43	0	0
7	0	5	0	0
8	0	7	0	0
9	0	2	0	0

CHAPTER 7

LATCH DESIGN FOR TRANSIENT PULSE TOLERANCE

As an exercise in using FAST, and as an interesting problem in its own right, the design space of DFFs is explored in this chapter to see what kind of tradeoffs are possible between transient pulse tolerance and performance. In the course of designing transient pulse tolerant DFFs, TIFAS is used in every step to determine the limits of the pulse tolerance needed, showing its value in a fault tolerant design [58]. A transient pulse tolerant latch is different from a traditional SEU-hardened latch in the following way: a traditional SEU-hardened latch is designed to tolerate direct ion hits whereas a transient pulse tolerant latch is designed to tolerate transient pulses that arise in the combinational part of the circuit and propagate to the input of the latch. Since no work has been previously done on transient pulse tolerant latches, we are limited to examining the SEU-hardened latch designs.

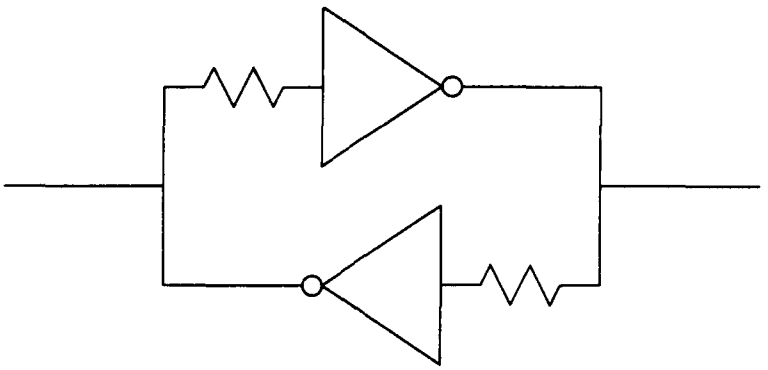


Figure 7.1 A hardened latch design which uses resistors in the feedback path

7.1 Radiation Hardened Latch Designs

Although the topology of the radiation hardened latches varies, the fundamental concept is the same: provide data redundancy in duplicate storage nodes which can correct the SEU.

The most common technique for hardening latches is to use resistors in the feedback path as shown in Figure 7.1 [59]. The idea here is that with the proper choice of the resistances, any SEU occurring at the drain nodes will die out before it has had time to propagate around the feedback loop. We may also view the resistors as providing redundant storage nodes which can hold uncorrupted data for a certain length of time. A major drawback of this technique is the performance penalty introduced by the resistors. The following three techniques require less performance overhead.

In the paper by Rockett [60], a radiation hardened latch design is described which uses extra transistors to provide the data redundancy. When the clock line is high, the redundant transistors are disabled to allow the latch to be written in a normal manner.

When the clock line is low, however, the redundant transistors maintain the integrity of the latch node holding the high value, making the latch SEU tolerant.

Weaver et al. uses voltage division to achieve upset protection in one direction [61]. Resistors are inserted between the drains of PMOS and NMOS transistors of an inverter. If the feedback point is at the drain of the NMOS transistor, an ion-hit-induced voltage pulse at the drain of the PMOS transistor will be voltage divided before reaching the input of the feedback inverter. With the proper choice of the resistances and the proper choice of feedback points, the latch can be made immune to one type of upset. That is, it can be made immune to either the 0 to 1 flip or 1 to 0 flip. If the latch is immune to the 1 to 0 flip, full SEU immunity can be achieved by duplicating the latch and connecting the outputs to an AND gate.

The last technique we will look at was proposed by Liu and Whitaker [62]. Unlike the technique by Rockett, the latch here is designed from the ground up with redundancy in mind. The latch consists of four storage nodes. Only transistors of one type are connected to a storage node, making the nodes immune to either the 0 to 1 flip or 1 to 0 flip. The nodes are fed back with the above immunity in mind such that the uncorrupted node storing the same information can always correct the corrupted node. Unfortunately, none of these techniques can be directly applied to guard against transient pulses at the input.

7.2 Transient Pulse Tolerant Latch Design

For the SEU tolerant latches discussed above, it is possible to incur only a small performance overhead through the use of information redundancy in space. The use of space redundancy is possible because uncorrupted data can easily be made accessible by the corrupted part of the latch. However, for transient pulse tolerant designs, space redundancy is not an option because uncorrupted data are not readily available at the time the transient pulse occurs. The only source of uncorrupted data is from a duplicate circuit, and duplicating the whole circuit is a very expensive proposition. Since information redundancy must come from either space or time, the only remaining option is information redundancy in time, and this implies a significant performance penalty.

Although performance penalty is unavoidable, we should still examine the tradeoffs that can be made. As the performance becomes worse, the transient pulse tolerance should improve. For each circuit and each application, the degree of transient pulse tolerance needed will be different. In this section, we find, through SPICE simulations, the most effective resistor to insert for obtaining transient pulse tolerance, and in the next section, we examine the tradeoff between performance and transient pulse tolerance.

For this design, the modified *dfnf311* DFF whose latching windows are shown in Table 3.3 has been used as the base flip-flop. The input stage and the master latch of the flip-flop are shown in Figure 7.2. The transistors C1-C4 are used to buffer the clock input. The transistors I1-I4 form the input inverter and pass gate combination. The transistors L1-L6 form the back-to-back inverters and pass gate combination. The

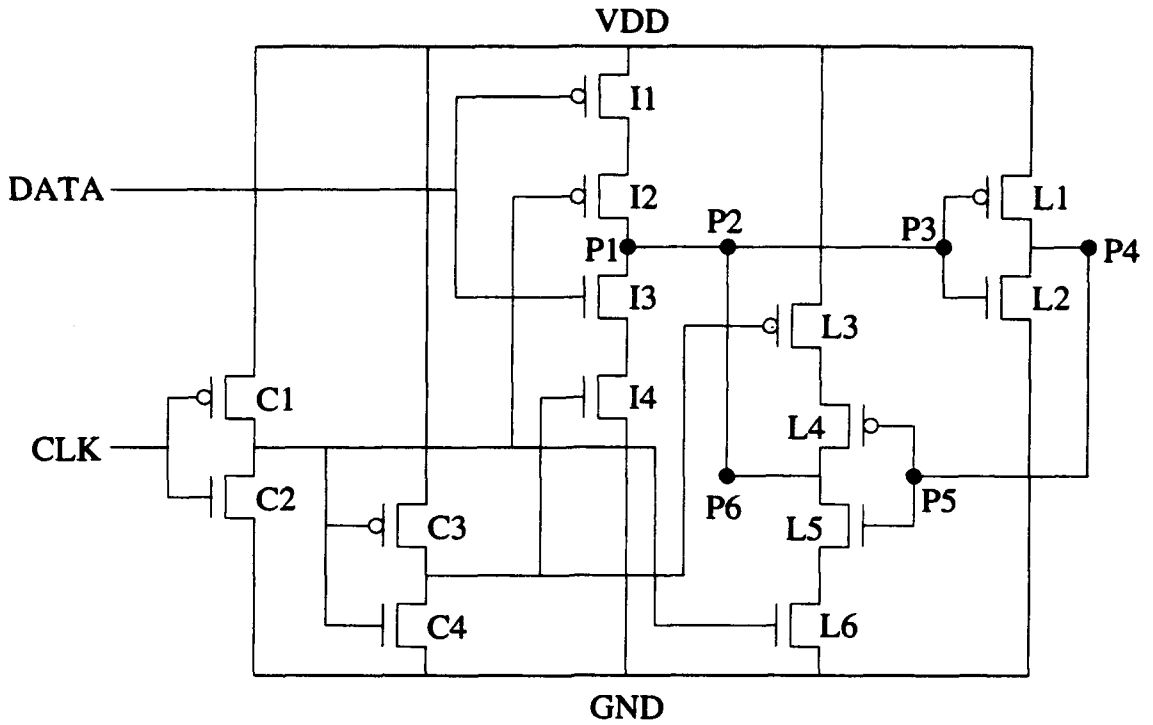


Figure 7.2 *Dfnf311* D flip-flop input stage and master latch used in this thesis

functionally equivalent circuit is shown in Figure 7.3. Transistors I1 and I2 have wider widths and transistors L5 and L6 have narrower widths than the original, and the final dimensions used are shown in Table 7.1.

Table 7.1 Base DFF transistor dimensions

Transistor	W/L	Transistor	W/L
C1	23/2	C2	14/2
C3	23/2	C4	14/2
I1	45/2	I2	45/2
I3	15/2	I4	15/2
L1	22/2	L2	15/2
L3	9/2	L4	9/2
L5	4/2	L6	4/2

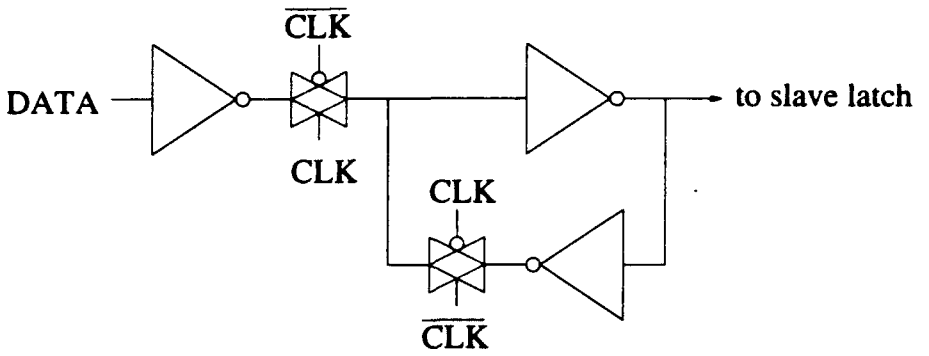


Figure 7.3 *Dfnf311* Functionally equivalent circuit for the D flip-flop input stage and master latch

To make the DFF transient pulse tolerant, the input stage has to be slow in order to filter out the pulse. A resistor can be inserted between the points P1 and P2 in Figure 7.2 to slow down the input stage. With the proper choice of the resistance, the latching windows can be made to vanish, hardening the DFF against transient pulses at the input.

Other options for resistor insertion are between points P2 and P3, P2 and P6, and P4 and P5. The reason these resistors should be considered is that they are in the feedback path of the latch. It is possible that the feedback mechanism can help to eliminate or reduce the effects of the transient pulse, similar to the effect resistors in the feedback path have for radiation hardened latches.

The resistor between points P2 and P6 has been found to be ineffective because this resistor does not help in any way to reduce the effects of the transient pulse which arrives at point P1. The resistor between points P4 and P5 has been found to be ineffective as well. On first thought, with proper transistor sizing and proper resistance between P4 and P5, we may think the transistors L3-L6 can help to reduce the effects of the transient

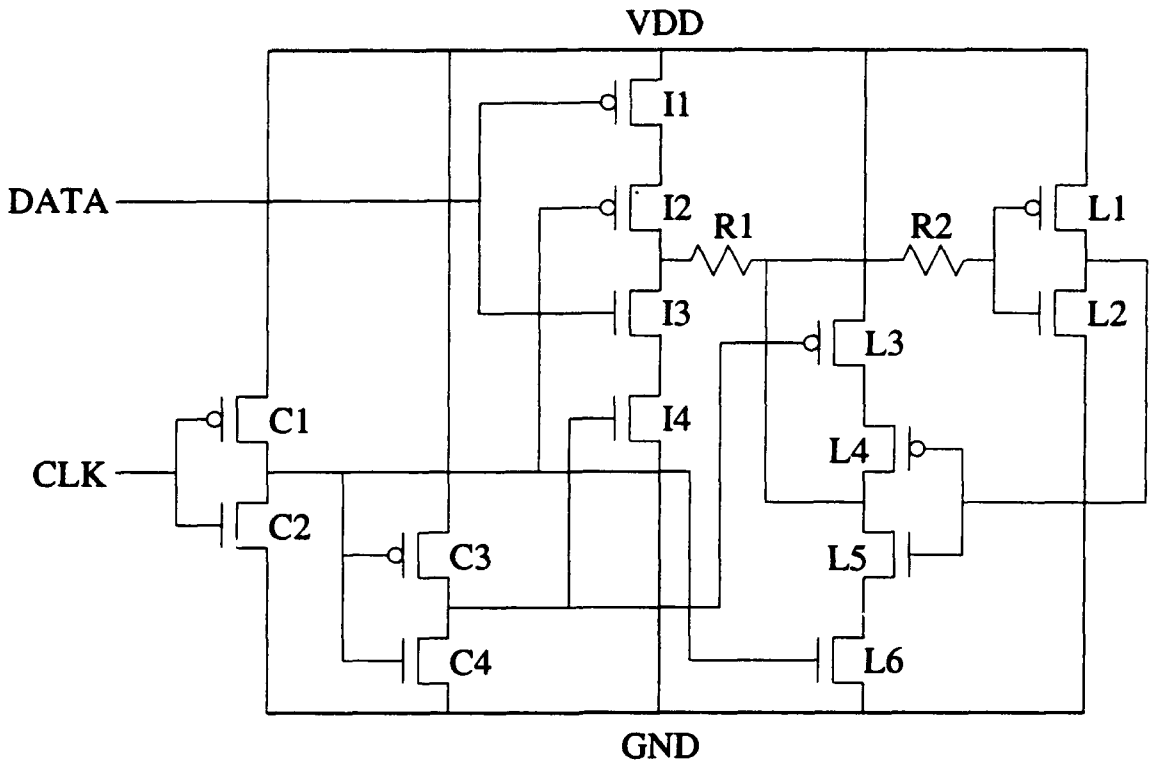


Figure 7.4 *Dfnf311* D flip-flop with possible places for adding resistors R1 and R2

pulse arriving at P1. However, this is not straightforward since the feedback mechanism might interfere with the write operation. Furthermore, this has only a limited impact in any case since L3-L6 only turn on when the clock line goes low. Therefore, the only cases we should consider are these: adding a resistance between points P1 and P2, and between points P2 and P3. From here on, the resistor between P1 and P2 will be referred to as R1, and the resistor between points P2 and P3 will be referred to as R2, as shown in Figure 7.4.

Table 7.2 and Table 7.3 show the latch windows for adding a 10 kΩ R1 and 10 kΩ R2, respectively. Comparing the tables, we can see that R1 is more effective in reducing the latching window. However, it is possible that R1 might incur more performance penalty

than R2, so the relative performances of the two resistors should be compared in terms of the setup time required.

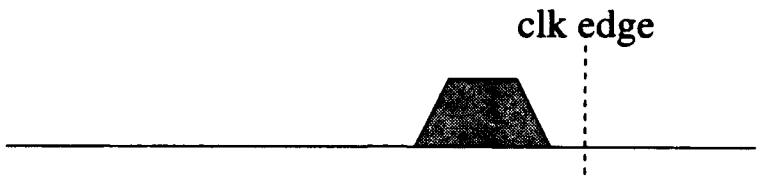
Table 7.2 *Dfnf311* latching windows with R1 = 10 $k\Omega$ (in ns)

Pulse Duration	1-0-1 Pulse		0-1-0 Pulse	
	Earliest Time	Latest Time	Earliest Time	Latest Time
1.3	–	–	–	–
1.4	0.0	0.2	–	–
1.5	-0.2	0.4	–	–
1.6	-0.2	0.6	–	–
1.7	-0.3	0.7	0.1	0.3
1.8	-0.4	0.8	0.1	0.5
1.9	-0.4	0.9	0.0	0.7
2.0	-0.5	1.0	-0.1	0.8

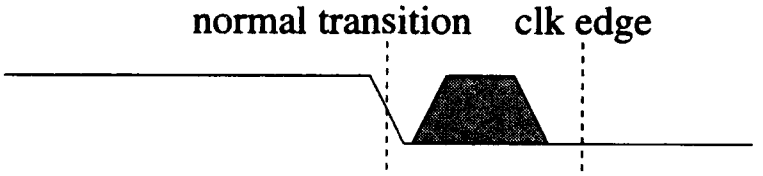
Table 7.3 *Dfnf311* latching windows with R2 = 10 kΩ (in ns)

Pulse Duration	1-0-1 Pulse		0-1-0 Pulse	
	Earliest Time	Latest Time	Earliest Time	Latest Time
1.0	–	–	–	–
1.1	0.2	0.6	0.2	0.8
1.2	0.2	0.7	0.1	0.9
1.3	0.2	0.8	0.0	1.0
1.4	0.1	0.9	0.0	1.1
1.5	0.1	1.0	-0.1	1.2
1.6	0.1	1.1	-0.1	1.3
1.7	0.0	1.2	-0.1	1.4
1.8	0.0	1.3	-0.2	1.5
1.9	0.0	1.4	-0.2	1.6
2.0	0.0	1.5	-0.2	1.7

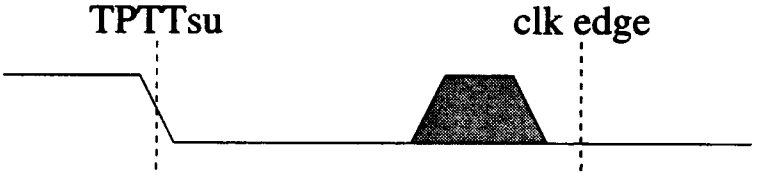
Normally, setup time is defined as the time with respect to the clock edge at which the input signal should be stable in order to guarantee proper latch operation. With the transient pulse tolerant latch, we have a somewhat different problem. The latch windows



(a) pulse is outside the latching window



(b) normal transition near the pulse causes it to become latched



(c) transient pulse tolerant setup time such that it isn't latched

Figure 7.5 Illustration of the concept of $TPTT_{su}$

above were found assuming that the input signal has been stable for a long time prior to the arrival of the transient pulse, as shown in Figure 7.5(a). If, however, a normal transition occurs somewhat near the clock edge, this might cause the latching window to become larger. For example, if a transition on the DFF input occurs near the clock edge and the same pulse arrives as shown in Figure 7.5(b), this pulse may be latched in reality although it is outside the latching window. Therefore, the transition in effect enlarges the latching window. What we need to do is to define a setup time as shown in Figure 7.5(c) that will preserve the latching window size.

Definition 1 $\mathrm{TPTT}_{su}(pw)$ *is the transient pulse tolerant setup time at which point the input signal to DFF should be stable in order to guarantee the same latching window size for a pulse width of pw as in the case where the input signal has been stable for an infinitely long time.*

For R1, $TPTT_{su}(1.0ns) = -3.5$ ns and for R2, $TPTT_{su}(1.0ns) = -4.7$ ns. Not only does R1 provide for a smaller latching window for a given resistance, but also require shorter setup time. Therefore, inserting R1 is the most effective choice among the alternatives we have looked at. As we increase the value of R1, the latching windows will decrease, but the $TPTT_{su}$ will increase, worsening the performance. In the next section, we examine in detail the tradeoff between transient pulse tolerance and the performance penalty.

7.3 Fault Tolerance and Performance Tradeoff

7.3.1 Experimental Setup

For the experiment conducted to compare the tradeoff between transient pulse tolerance and speed, a set of circuits from ISCAS-89 suite of sequential benchmark circuits has been chosen. The gates are from the standard cell library from Mississippi State University. Gate delays and latching windows were obtained using SPICE3 with level 1 models, similar to the experiments conducted in the previous chapter.

In each of the circuits simulated, 100,000 randomly chosen faults were injected. The injected faults were chosen randomly over time, space, and the amount of charge injected.

```
Cell library:      standard cell library from Mississippi State University
SPICE model:       level 1
Number of faults injected:  100,000 per circuit
Fault injection node:    randomly chosen over all nodes in the circuit
                         excluding input, output and DFF output nodes
Total charge injected:   randomly chosen from 1 pC to 9 pC
Fault injection time:    randomly chosen over the whole clock period
Fault injection clock cycle:  evenly distributed over the middle
                              portion of the simulation run
Input vector suite:      automatically generated from
                         Sequential Circuit Test Generator (STG)
```

Figure 7.6 The experimental setup for finding the tradeoff between transient pulse tolerance and speed

The charge injected is evenly distributed from 1 pC to 9 pC in 1 pC increments. Faults were injected at the output nodes of all gates, except at the DFF output nodes and primary output nodes. DFF output nodes were excluded because there is a possibility of the injected fault directly flipping the DFF, which is a different phenomenon than the one we are interested in. Finally, the input vectors for exercising the circuits are from [57]. Figure 7.6 summarizes the experimental setup.

7.3.2 Experiment

Table 7.4 shows the distribution of the widths of the transient pulses latched into the flip-flops for the base *dfnf311* with R1 = 0 Ω. According to the table, most of the pulses that are latched are less than 2.0 ns long for the smaller circuits, and even for the larger circuits, most are less than 2.4 ns long. The 32 pulses in column $\geq$ 2.4 for the circuit

s35932 are in fact 2.4 ns long in duration. Therefore, we will concentrate on pulse widths that are less than or equal to 2.4 ns long.

Table 7.4 The distribution of the widths of the transient pulses latched into the flip-flops with R1 = 0 Ω and 100,000 fault injections

Circuit	Latched Faults	0.8-1.1	1.2-1.5	1.6-1.9	2.0-2.3	≥2.4
s208	1141	902	203	83	0	0
s641	409	410	45	7	0	0
s713	428	413	49	4	0	0
s820	568	219	286	58	4	2
s953	595	475	114	64	0	0
s1196	489	215	207	63	9	2
s1238	452	204	196	51	4	1
s1494	133	112	18	3	0	0
s5378	792	318	310	258	64	2
s35932	712	596	187	50	29	32

The $TPTT_{su}$ tends to be longer for a longer pulse width. For some of the smaller circuits, longer transient pulses do not occur. However, for the sake of simplicity, we will only look at $TPTT_{su}(2.4ns)$, even when it is being applied to the smaller circuits. Table 7.5 summarizes the $TPTT_{su}(2.4ns)$ for various values of the resistor R1.

Table 7.5 The $TPTT_{su}(2.4ns)$ values for various R1 values (in ns)

	0 Ω	5 kΩ	10 kΩ	15 kΩ	20 kΩ
$TPTT_{su}(2.4ns)$	-1.7	-3.8	-7.5	-7.8	-9.4

The transient fault simulator FAST was used to simulate 100,000 fault injections per circuit with the different latching windows corresponding to the various values of R1. Table 7.6 summarizes the results. The normal setup time with R1 = 0 Ω is -0.3 ns. As the table shows, the R1 value of 15 kΩ with 7.5 ns penalty compared to the normal setup time is seen to tolerate most of the transient pulses. Even the R1 value of 10 kΩ

decreased the likelihood of latching a transient pulse by one to two orders of magnitude compared to no R1, but the performance penalty is about the same as that of 15 kΩ. Since the transient pulse width distribution is different for each circuit, the value of R1 should be optimized for each circuit individually for better performance.

The penalty of 7.5 ns should be compared with respect to the clock period used. As shown in Table 6.2, the clock periods used for these circuits vary from 5.9 ns to 19.3 ns. The penalty of 7.5 ns represents between 39 and 127% performance overhead with respect to the clock cycle used. Although this penalty is substantial, it may be acceptable for critical systems.

Table 7.6 The number of latched faults out of 100,000 injections in ISCAS-89 benchmark circuits with various R1 values

Circuit	0 Ω	5 kΩ	10 kΩ	15 kΩ	20 kΩ
s208	1141	242	81	1	0
s641	409	67	3	0	0
s713	428	65	1	0	0
s820	568	260	28	2	0
s953	595	123	10	0	0
s1196	489	267	70	6	0
s1238	452	239	68	3	1
s1494	133	14	1	0	0
s5378	792	450	187	54	1
s35932	712	73	4	1	0

A couple of observations are in order. It was found that small imbalances in 0-1-0 pulse and 1-0-1 pulse latching windows were magnified with the insertion of R1. Furthermore, the $TPTT_{su}$ times for the different pulse types differed by as much as several nanoseconds. This indicates that special care must be taken to balance the 0-1-0 pulse and the 1-0-1 pulse latching behaviors prior to R1 insertion to maximize performance.

The second observation is that the transient pulse width can be decreased. Since the pulse width is a function of the transistor strength and the node capacitance, using larger transistors or cutting down on the number of fanout gates is likely to reduce the pulse width. With reduced pulse widths, a smaller value of R1 can be used; therefore, performance can be improved.

Another interesting aspect to examine is the DFF flip distribution, which is important to researchers injecting latch flip errors in high level simulators as mentioned in the previous chapter. Table 7.7 shows the flip distributions for various values of R1. While it is clear that fewer faults become latched as the R1 value is increased, it is not so obvious what happens with the flip distributions. In some cases, the distribution becomes more widely spread with increasing R1 while in others the distribution becomes sharper.

Table 7.7 The DFF flip distributions for various values of R1, with 100,000 fault injections for each case (in percent)

	R1 = 0 Ω			**R1 = 5 kΩ**			**R1 = 10 kΩ**		
Circuit	**1**	**2**	**≥3**	**1**	**2**	**≥3**	**1**	**2**	**≥3**
s208	97	2	1	90	6	4	84	12	4
s641	89	9	2	87	10	3	100	0	0
s713	92	8	0	88	12	0	100	0	0
s820	100	0	0	100	0	0	100	0	0
s953	90	10	0	97	3	0	100	0	0
s1196	99	1	0	100	0	0	100	0	0
s1238	99	1	0	99	1	0	100	0	0
s1494	100	0	0	100	0	0	100	0	0
s5378	91	5	4	92	6	2	92	8	1
s35932	99	0	1	88	0	12	0	0	100

In this exercise of designing a transient pulse tolerant DFF, the value of TIFAS has been clearly demonstrated. Without it, it would have been very difficult to have an

idea of the kind of pulse duration the DFF should be designed to tolerate, let alone the tradeoff possible between performance and transient pulse tolerance.

CHAPTER 8

CONCLUSIONS

The work in this thesis was motivated by the lack of efficient tools for simulating transient fault injections in the combinational logic portion of a sequential circuit. To address this problem, a gate-level simulation tool with proper modeling has been developed. This tool, called TIFAS, represents an effective compromise between accuracy and speed. TIFAS is shown to be between four and five orders of magnitude faster than a very accurate circuit simulator with some loss in accuracy for the simulation of standard cell based fully static CMOS synchronous sequential circuits.

To further speed up the entire simulation, another simulator, TPROOFS, is used once a transient fault becomes latched into one or more DFFs. The reason for the use of TPROOFS is that once a fault is latched, the circuit can be treated as pure logic without any timing information. For the simulation of pure logic, a zero delay parallel simulator such as TPROOFS is more efficient than TIFAS.

The combination of TIFAS and TPROOFS has been used on ISCAS-89 benchmark circuits both to show the capability of the tools and to analyze the circuits. The experiments show that the one-bit flip model is not a good model for injecting faults in

highly fault tolerant systems. A better model would include multiple DFF flips with the appropriate probabilities derived from TIFAS simulations.

The experiments also show that at the pin level, a simple model cannot be found in general, at least for the alpha-particle-induced or other transient pulse-induced errors.

The usefulness of TIFAS is also demonstrated in the design of a transient pulse tolerant DFF. With the insertion of a resistor immediately after the input stage in the master latch of the DFF, the DFF can be made to filter out transient pulses, and TIFAS is used to examine the tradeoff between performance penalty and transient pulse tolerance of this DFF design.

8.1 Future Research

There are several extensions that can be made to improve the performance and accuracy of TIFAS and also fault injection experiments in general. In TIFAS, for example, better models can be used. The current implementation uses only one gate level of pulse width modeling. This can be extended to two, which will aid in improving the accuracy of the simulator. Also, the gate-delay models can be made to reflect the input slew rates, which will result in more accurate delays and also more accurate pulse widths. Also in TIFAS, to speed up the simulator, more static analysis can be done to remove the faults that occur far outside the latching windows by using the minimum and maximum delays from each node to all of the DFFs.

Although FAST provides quick and reasonable DFF flip distributions in fault injection experiments, it is not the answer to all fault injection problems. More accurate simulations are needed for a more detailed analysis of the circuit, and the only way to obtain more accurate answers is to use a more accurate simulator, be it ILLIADS or SPICE. TIFAS can help in this endeavor as well.

First, TIFAS may be modified to obtain a superset or near-superset of all the faults that would be latched by SPICE. A possible solution involves the following. First, the pulse width model may be modified to give widths that are at least as wide as the ones reported by SPICE. Second, the latching window models may also be modifed to give worst-case windows. Finally, minimum and maximum delays for each gate may be used to see if the pulse arrival time range at a DFF overlaps with the latching window. If it does, it is marked latched. This approach would report most, if not all, of the faults that would be latched by SPICE, as long as the minimum and maximum delays are modeled with care. Once the superset is obtained, SPICE or ILLIADS may be used to simulate only those faults. Obviously, there will be significant savings in time.

Another approach that may be used is to obtain a set of activated gates in TIFAS and to simulate only those gates in SPICE. For example, if a logical path exists between the fault injected node and a DFF, all of the gates on the path as well as all of the gates which influence the voltage waveforms at any of the nodes on the path may be extracted to be simulated in SPICE. This approach has the potential of large savings in simulation time since SPICE uses an n^2 to n^3 algorithm, where n is the number of nodes in the

circuit. Furthermore, the two approaches outlined above can be combined to obtain an even better speedup in both ILLIADS and SPICE simulations.

For the best accuracy in the shortest possible simulation time, all three simulators, TIFAS, ILLIADS and SPICE, can be used. First, TIFAS may be used to obtain a superset as proposed above. Then, ILLIADS may be used to filter out some of the faults which will not be latched in SPICE. Finally, SPICE may be used on the remaining faults. The inaccuracy of ILLIADS can be compensated for by simulating the same fault several times in ILLIADS with slightly different injection times. This approach would still be faster than using SPICE alone since ILLIADS is much faster.

REFERENCES

[1] H. Ball and F. Hardy, "Effects and detection of intermittent failures in digital systems," in *FJCC, AFIPS Conf. Proc.*, vol. 35, 1969, pp. 329-335.

[2] R. Iyer and D. Rossetti, "A measurement-based model for workload dependence of CPU errors," *IEEE Trans. Comput.*, vol. C-35, pp. 511-519, June 1986.

[3] T. C. May and M. H. Woods, "Alpha-particle-induced soft errors in dynamic memories," *IEEE Trans. Electron Devices*, vol. ED-26, no. 1, pp. 2-9, Jan. 1979.

[4] J. C. Pickel and J. T. Blandford, Jr., "Cosmic-ray-induced errors in MOS devices," *IEEE Trans. Nucl. Sci.*, vol. NS-27, no. 2, pp. 1006-1015, April 1980.

[5] J. H. Adams, Jr., "The variability of single event upset rates in the natural environment," *IEEE Trans. Nucl. Sci.*, vol. NS-30, no. 6, pp. 4475-4480, Dec. 1983.

[6] E. L. Peterson, J. B. Langworthy and S. E. Diehl, "Suggested single event upset figure of merit," *IEEE Trans. Nucl. Sci.*, vol. NS-30, no. 6, pp. 4533-4539, Dec. 1983.

[7] K. C. Holland and J. G. Tront, "Probability of latching single event upset errors in VLSI circuits," *IEEE Proc. of Southeast Conf.*, pp. 109-113, 1991.

[8] D. Chlouber, P. O'Neill and J. Pollock, "General upper bound on single-event upset rate," *IEEE Trans. Nucl. Sci.*, vol. 37, no. 2, pp. 1065-1071, April 1990.

[9] A. Campbell, P. McDonald and K. Ray, "Single event upset rates in space," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 6, pp. 1828-1835, Dec. 1992.

[10] C. I. Underwood, J. W. Ward, C. S. Dyer and A. J. Sims, "Observations of single-event upsets in non-hardened high-density SRAMs in sun-synchronous orbit," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 6, pp. 1817-1827, Dec. 1992.

[11] R. Koga, W. R. Crain, K. B. Crawford, S. J. Hansel, S. D. Pinkerton and T. K. Tsubota, "The impact of ASIC devices on the SEU vulnerability of space-borne computers," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 6, pp. 1685-1692, Dec. 1992.

[12] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electron. Res. Lab., Univ. of California, Berkeley, Rep. ERL-M520, May 1975.

[13] H. Cha, E. M. Rudnick, G. S. Choi, J. H. Patel, and R. K. Iyer, "A fast and accurate gate-level transient fault simulation environment," *Digest, 23th Int. Symp. Fault-Tolerant Comput.*, June 1993, pp. 310-319.

[14] T. M. Niermann, W. Cheng, and J. H. Patel, "PROOFS: a fast, memory-efficient sequential circuit fault simulator," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 2, pp. 198-207, Feb. 1992.

[15] H. Cha and J. H. Patel, "A logic-level model for α-particle hits in CMOS circuits," *Proc. Int. Conf. Computer Design*, Oct. 1993, pp. 538-542.

[16] A. Dharchoudhury, S. M. Kang, H. Cha and J. H. Patel, "Fast timing simulation of transient faults in digital circuits," to appear in *Proc. Int. Conf. Comput.-Aided Design*, 1994.

[17] R. K. Iyer and D. Tang, "Experimental analysis of computer system dependability," Tech. Rept. UILU-ENG-93-2227, Univ. of Illinois, Champaign-Urbana, Sept. 1993.

[18] J. Cusick, R. Koga, W. A. Kolasinski and C. King, "SEU vulnerability of the Zilog Z-80 and NSC-800 microprocessors," *IEEE Trans. Nucl. Sci.*, vol. NS-32, no. 6, pp. 4206-4211, Dec. 1985.

[19] J. Karlsson, U. Gunneflo, and J. Torin, "The effects of heavy-ion induced single event upsets in the MC6809E microprocessor," in *Proc. 4th Int. Conf. Fault-Tolerant Comput. Syst.*, GI/ITG/GMA, Baden, W. Germany, 1989.

[20] U. Gunneflo, J. Karlsson, J. Torin, "Evaluation of error detection schemes using fault injection by heavy-ion radiation," *Digest, 19th Int. Symp. Fault-Tolerant Comput.*, June 1989, pp. 340-347.

[21] T. Takano, T. Yamada, K. Shutoh and N. Kanekawa, "Fault-tolerance experiments of the 'Hiten' onboard space computer," *Digest, 21st Int. Symp. Fault-Tolerant Comput.*, June 1991, pp. 26-33.

[22] J. Lala, "Fault detection isolation and reconfiguration in FTMP: Methods and experimental results," in *Proc. 5th AIAA/IEEE Digital Avion. Syst. Conf.*, Nov. 1983, pp. 21.3.1-21.3.9.

[23] M. A. Schuette and J. P. Shen, "Processor control flow monitoring using signatured instruction streams," *IEEE Trans. Comput.*, vol. C-36, no. 3, pp. 264-275, Mar. 1987.

[24] J. Arlat, Y. Crouzet and J.-C. Laprie, "Fault injection for dependability validation of fault-tolerant computing systems," *Digest, 19th Int. Symp. Fault-Tolerant Comput.*, June 1989, pp. 348-355.

[25] R. Chillarege and N. S. Bowen, "Understanding large system failures - a fault injection experiment," *Digest, 19th Int. Symp. Fault-Tolerant Comput.*, June 1989, pp. 356-363.

[26] J. H. Barton, E. W. Czeck, Z. Z. Segall and D. P. Siewiorek, "Fault injection experiments using FIAT," *IEEE Trans. Comput.*, vol. 39, no. 4, pp. 575-582, April 1990.

[27] K. W. Li, J. R. Armstrong and J. G. Tront, "An HDL simulation of the effects of single event upsets on microprocessor program flow," *IEEE Trans. Nucl. Sci.*, vol. NS-31, no. 6, pp. 1139-1144, Dec. 1984.

[28] M. Rimén and J. Ohlsson, "A study of the error behavior of a 32-bit RISC subjected to simulated transient fault injection," in *Proc. Int. Test Conf.*, Nov. 1992, pp. 696-704.

[29] E. W. Czeck and D. P. Siewiorek, "Effects of transient gate-level faults on program behavior," in *Digest, 20th Int. Symp. Fault-Tolerant Comput.*, June 1990, pp. 236-243.

[30] R. McPartland, "Circuit simulations of alpha-particle-induced soft errors in dynamic RAM's," *IEEE J. Solid-State Circuits*, vol. SC-16, no. 1, pp. 31-34, Feb. 1981.

[31] G. C. Messenger, "Collection of charge on junction nodes from ion tracks," *IEEE Trans. Nucl. Sci.*, vol. NS-29, no. 6, pp. 2024-2031, Dec. 1982.

[32] R. Johnson, S. Diehl-Nagle and J. Hauser, "Simulation approach for modeling single event upsets on advanced CMOS SRAMS," *IEEE Trans. Nucl. Sci.*, vol. NS-32, no. 6, pp. 4122-4127, Dec. 1985.

[33] G. S. Choi, R. K. Iyer and D. G. Saab, "Fault behavior dictionary for simulation of device-level transients," *Proc. Int. Conf. Comput.-Aided Des.*, 1993, pp. 6-9.

[34] G. L. Ries, G. S. Choi and R. K. Iyer, "Device-level transient fault modeling," *Digest, 24th Int. Symp. Fault-Tolerant Comput.*, June 1994, pp. 86-94.

[35] R. Saleh, "Nonlinear relaxation algorithms for circuit simulation," Memorandum no. UCB/ERL M87/21, Electron. Res. Lab., Univ. of California, Berkeley, 1987.

[36] E. L. Acuna, J. P. Dervenis, A. J. Pagones, F. L. Yang, and R. A. Saleh, "Simulation techniques for mixed analog/digital circuits," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 353-363, Apr. 1990.

[37] G. Choi, R. K. Iyer, R. Saleh, and V. Carreno, "A fault behavior model for an avionic microprocessor: A case study," in *Int. Working Conf. Dependable Computing for Critical Applications*, Aug. 1989, pp. 71-77.

[38] F. L. Yang and R. A. Saleh, "Simulation and analysis of transient faults in digital circuits," *IEEE J. Solid-State Circuits*, vol. 27, no. 3, pp. 258-264, Mar. 1992.

[39] G. S. Choi and R. K. Iyer, "FOCUS: an experimental environment for fault sensitivity analysis," *IEEE Trans. Comput.*, vol. 41, no. 12, pp. 1515-1526, Dec. 1992.

[40] Y.-H. Shih, Y. Leblebici and S. M. Kang, "ILLIADS: a fast timing and reliability simulator for digital MOS circuits," *IEEE Trans. Comput.-Aided Des.*, vol. 12, no. 9, pp. 1387-1402, Sept. 1993.

[41] V. A. Carreno, G. Choi, and R. K. Iyer, "Analog-digital simulation of transient-induced logic errors and upset susceptibility of an advanced control system," NASA Technical Memorandum 4241, Nov. 1990.

[42] OCTTOOLS-5.1 Part 1: User Guide, A. Casotto: Ed., Electron. Res. Lab., Univ. of California, Berkeley, Oct. 1991.

[43] T. L. Quarles, "SPICE3 version 3C1 users guide," Memorandum no. UCB/ERL M89/46, Electron. Res. Lab., Univ. of California, Berkeley, Apr. 1989.

[44] J. P. Uyemura, *Fundamentals of MOS Digital Integrated Circuits.* Reading, MA; Addison-Wesley, 1988.

[45] D. Overhauser, I. N. Hajj, and V. B. Rao, "Switch-level timing analysis of VLSI MOS circuits including parasitics," *Proc. IEEE Int. Symp. Circuits Systems,* May 1986, pp. 761-764.

[46] N. Hedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Trans. Comput.-Aided Des.*, vol. CAD-6, no. 2, pp. 270-281, Mar. 1987.

[47] M. Shoji, *CMOS Digital Circuit Technology.* Englewood Cliffs, NJ; Prentice Hall, 1988.

[48] S. R. Vemuru and A. R. Thorbjornsen, "A model for delay evaluation of a CMOS inverter," *Proc. IEEE Int. Symp. Circuits Systems*, May 1990, pp. 89-92.

[49] T. J. Chaney and C. E. Molnar, "Anomalous behavior of synchronizer and arbiter circuits," *IEEE Trans. Comput.*, vol. C-22, no. 4, pp. 421-422, Apr. 1973.

[50] G. R. Couranz and D. F. Wann, "Theoretical and experimental behavior of synchronizers operating in the metastable region," *IEEE Trans. Comput.*, vol. C-24, no. 6, pp. 604-616, June 1975.

[51] H. J. M. Veendrick, "The behavior of flip-flops used as synchronizers and prediction of their failure rate," *IEEE J. Solid-State Circuits*, vol. SC-15, no. 2, pp. 169-176, Apr. 1980.

[52] P. A. Stoll, "How to avoid synchronization problems," *VLSI Design*, pp. 56-59, Nov. 1982.

[53] J. H. Hohl, W. R. Larsen and L. C. Schooley, "Prediction of error probabilities of integrated digital synchronizer," *IEEE J. Solid-State Circuits,* vol. SC-19, no. 2, pp. 236-244, Apr. 1984.

[54] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. Int. Symp. Circuits Systems*, May 1989, pp. 1929-1934.

[55] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design.* New York, NY; Computer Science Press, 1990.

[56] *Databook: LCB15 Cell-Based ASIC Macrocells, Macrofunctions*, LSI Logic Corporation, Mar. 1988.

[57] W. -T. Cheng and S. Davidson, "Sequential circuit test generator (STG) benchmark results," *Proc. Int. Symp. Circuits Systems*, May 1989, pp. 1939-1941.

[58] H. Cha and J. H. Patel, "Latch design for transient pulse tolerance," to appear in *Proc. Int. Conf. Comput. Des.*, Oct. 1994.

[59] J. Andrews, J. Schroeder, B. Gingerich, W. Kolasinski, R. Koga and S. Diehl, "Single event error immune CMOS RAM," *IEEE Trans. Nucl. Sci.*, vol. NS-29, no. 6, pp. 2040-2043, Dec. 1982.

[60] L. R. Rockett, Jr., "An SEU-hardened CMOS data latch design," *IEEE Trans. Nucl. Sci.*, vol. 35, no. 6, pp. 1682-1687, Dec. 1988.

[61] H. T. Weaver, W. T. Corbett, and J. M. Pimbley, "Soft error protection using asymmetric response latches," *IEEE Trans. Electron Devices*, vol. 38. no. 6, pp. 1555-1557, June 1991.

[62] M. Norley Liu and Sterling Whitaker, "Low power SEU immune CMOS memory circuits," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 6, pp. 1679-1684, Dec. 1992.

VITA

Hungse Cha was born in Seoul, Korea, on July 21, 1966. He received the B.S. degree in Electrical Engineering from the California Institute of Technology in 1988, and the M. S. degree in Electrical Engineering from the University of Illinois, Urbana, Illinois, in 1990. He was a teaching assistant from 1988 to 1989 at the University of Illinois. Since 1989, he has been with the Coordinated Science Laboratory at the University of Illinois as a research assistant working on various projects. His research interests include VLSI design for reliability and fault-tolerance, analog circuit design, testing and fault simulation. After completing his doctoral dissertation, he will be joining Hewlett-Packard Company in Cupertino, CA.